LNCS

LNC

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

293

Carl Pomerance (Ed.)

Advances in Cryptology — CRYPTO '87

Proceedings

Springer-Verlag

Berlin Heidelberg New York London Paris Tokyo

Editor

Carl Pomerance
Department of Mathematics, The University of Georgia
Athens, Georgia 30602, USA

CR Subject Classification (1987): E.3

ISBN 3-540-18796-0 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-18796-0 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210

Preface

This book is the proceedings of CRYPTO'87, one in a series of annual conferences devoted to cryptologic research. For citations of proceedings of CRYPTO and Eurocrypt conferences before 1986, see

Advances in Cryptology-CRYPTO'86 Proceedings, A. M. Odlyzko, ed., Lecture Notes in Computer Science #263, Springer, 1987.

Papers in this volume are organized into seven sections. The first six sections comprise all of the papers on the regular program, including two papers on the program that unfortunately were not presented at the meeting. The seventh section contains some of the papers presented at the "Rump Session" organized by W. Diffie and also includes a short note by T. R. N. Rao which comments on the paper of R. Struik and J. van Tilburg.

CRYPTO'87 was attended by 170 people representing 19 countries. Responsible not only for the conference as a whole, G. B. Agnew also took care of local arrangements in Santa Barbara. We all owe him a debt of gratitude for his highly successful efforts.

It is my special pleasure to thank my fellow members of the Program Committee: T. A. Berson, E. F. Brickell, A. M. Odlyzko, and G. J. Simmons. They all were most prompt, efficient, and willing to cheerfully compromise on disagreements. My task would have been hopeless without them.

I also would like to thank the authors and attendees who made CRYPTO'87 such a success. Special thanks are due to University of Georgia secretaries D. Byrd and P. Sisk and L. B. Montz at Springer for their help in the production of this volume.

Athens, Georgia

Carl Pomerance

CRYPTO'87

A Conference on the Theory and Applications of Cryptographic Techniques

held at the University of California, Santa Barbara,
through the cooperation of the
Computer Science Department

August 16-20, 1987

sponsored by:

The International Association for Cryptologic Research

in cooperation with

The IEEE Computer Society Technical Committee
On Security and Privacy

ORGANIZERS

TABLE OF CONTENTS

STANDARDS FOR DATA SECURITY - A CHANGE OF DIRECTION

Wyn L.Price
National Physical Laboratory
Teddington, Middlesex, UK

Standards for data security - the achievement of acceptable privacy and integrity in data communication and storage - have been in preparation for the last fourteen years, beginning with the US Data Encryption Standard (DES). The DES was adopted as a US federal standard (1) in 1977, followed by adoption as an ANSI standard (2) in 1981. Since 1980 work has been in progress to develop a corresponding International Standards Organisation (ISO) text. For most practical purposes the ISO text was identical with the ANSI text; the only significant departure was that the eight parity bits allocated to the key in the US standard were left unallocated in the ISO text. The responsible ISO body was at first Technical Committee 97 (information processing), Working Group 1, TC97/WG1, which was followed by Sub-Committee 20 (data cryptographic techniques) of TC97, TC97/SC20. In May 1986 a discussion, followed by a resolution, took place in TC97, meeting in Washington, as a result of which a reference was made to the central governing body of ISO on which all national member bodies are represented, ISO Council, to decide whether it was wise to proceed to publication of the ISO standard. The outcome of this reference was that ISO Council decided to abandon work on the DES as a potential international standard. The decision was taken very late in the process of preparing the standard text, publication had been imminent.

The chief argument advanced in favour of the decision was that adoption as a standard might encourage overdependence on the DES algorithm. It is well-known that the financial institutions (banks, building societies, savings and loan, etc.) make very widespread use of the algorithm and the value of their daily transactions protected by the algorithm must be very substantial. This offers a very attractive potential target for criminal cryptanalysts. It was evidently feared within ISO that publication of an ISO standard for data encipherment would increase the attractiveness of the target by influencing even more users to depend on the one algorithm.

No-one is seriously suggesting that the useful lifetime of the DES is already over, but it is felt that preparation must be made for that time when it is judged no longer safe for use in protecting transactions of significant value or sensitivity. Various interesting academic results have been obtained in research investigations of the DES (3,4,5), but none of the people or groups involved is yet seriously claiming that the DES is broken. No one is yet able to predict whether the algorithm will succumb first to an analytic attack or to an exhaustive search for a key.

In the US the wish to replace the DES with more appropriate algorithms has led to the establishment of the Commercial COMSEC Endorsement Program (CCEP). This was originally meant to cover two security classes, Type I solely for US federal use and Type II for US federal and domestic commercial use, with the DES being phased out after 1988. However, it seems that the US financial institutions have requested that the DES be continued for financial applications beyond this date; the American Bankers Association has let it be known that the request for extension of approval for the DES has been granted. Neither Type I nor Type II CCEP algorithms are to be exportable and cannot therefore ever be put forward as candidates for international standardisation, should this be considered again in the future; the algorithms wil not be published.

In ISO the approach is somewhat different. ISO has decided to adopt the principle of a register of encipherment algorithms as a means of encouraging some degree of diversification in choice of encipherment algorithms. Algorithms, published and unpublished, will be entered on the register. Unpublished algorithms may be represented by name, supplier(s), block size and key domain; possibly speeds of operation may be entered. Published algorithms may be represented by a reference to a full and formal description of the algorithm; it is quite conceivable that the DES itself will figure in the register. Suppliers of algorithms will be free to offer their algorithms to ISO for entry on the register. Such an entry, however, is unlikely to offer any indication of the strength of an unpublished algorithm; it will be a matter for exercise of user judgement in choosing an algorithm to decide whether an algorithm is suitable for the proposed use. The decision to set up the register was noted by SC20, meeting in Ottawa in April 1987; the first relevant action was to establish a work item which is aimed at setting down the rules under which the register will operate. It is hoped that the operational rules will become available sometime in 1988, though a formal starting date for the register has not yet been decided.

Until recently work was also in progress within ISO to prepare a standard text for the RSA public key cryptosystem; this was a simple statement of the algorithm, the text of an implementation in Pascal, some sample parameters and an indication of the rules to be applied in choosing key material. However, a recent decision within ISO is to discontinue all work on standards for data encipherment; the embargo on data encipherment standards is therefore extended beyond the DES and now embraces public key algorithms and all others. Working Group 2 of SC20 (SC20/WG2) had been preparing not only the text of an RSA standard but also a technical report surveying recent developments in work on public key cryptography (a first edition of this technical report was published some years ago); the secretariat of TC97 now advises that work on the parts of the technical report relating to encipherment algorithms should also be discontinued. Regarding the RSA algorithm, this too is now fairly widely used in protecting transaction processing. It therefore seems likely that it will be offered as a candidate entry on the algorithm register. There will then be a need for a definitive statement of the algorithm in a form to which reference can be made; the academic papers in which the idea was first disclosed and later discussed and elaborated are not suitable for this purpose. The form that the definitive statement will need to take - it cannot be a formal standard text - has yet to be worked out.

In view of the removal of all the work on encipherment standards one might wonder what was left for TC97/SC20 to do. In fact the work programme of this committee is now heavier and more extensive than it was prior to the recent decisions.

The work programme adopted in April 1987 concentrates on operation of the register of algorithms and on standards for data security applications, including modes of operation for data encipherment, enhancement of communication protocols with security capability and security management (including management of encipherment keys).

One of the early effects of the removal of the work on data encipherment standards was to hold up work on two other standards which were also in an advanced state of preparation. These were respectively for 64 bit block cipher modes of operation and for enhancement of the physical layer of OSI (Open Systems Interconnection) with encipherment capability. The modes of operation document needed little doing to it to take into account the decision not to publish the DES standard text; this was because the modes of operation text had already been deliberately written in a general way, so that it applied to all 64 bit block ciphers and not just the DES. A little more effort was needed to change the physical layer standard. Publication of the 64 bit block modes of operation standard can be expected without undue further delay. Advancement of the physical layer text to Draft International Standard may also take place soon.

Work is in hand to prepare a standard for modes of operation of a block cipher not restricted to 64 bit blocks; this should present little difficulty, using the 64 bit block text as a basis. Any work on security enhancement at the link layer of OSI is now in abeyance because it was agreed that the demand for a standard in this context has not been established. The need for security enhancement at three other layers of OSI, namely network (layer 3), transport (layer 4) and presentation (layer 6) is recognised and work is now going ahead to prepare standard texts for these functions; the words of the respective work items specify 'conditions for the practical operation of cryptographic protection' at the various layers. Also in the context of secure communications architecture we have a new work item on 'practical conditions for the Associated Control Service Element (ACSE) authentication'.

In the area of key management the work programme is now more detailed; separate items on management of keys for secret key algorithms using secret key techniques, for management of keys for secret key algorithms using public key techniques and for management of keys for public key algorithms using public key techniques have been established. A further work item is to define a register of public keys and its functionality (not to be confused with the register of algorithms); the public key register is a service which will be required for many practical implementations of public key cryptography.

Peer entity authentication has a prominent place in the work programme. Texts are being circulated for draft proposal voting on a method for peer entity authentication using secret key algorithms and for two methods using public keys with two- and three-way handshakes. Work on digital signatures is strongly represented, with items on digital signature with 'shadow' and with 'imprint' (respectively reflecting direct signature of data and signature of data through the medium of a hash function); methods of generating hash functions are also included.

An important issue that arose during 1986 was the division of responsibilities between the committees of ISO working on communication protocols and architectures and SC20 working on data security. It was recognised that there was a common area of interest between TC97/SC6 (layers 1 - 4 of OSI), TC97/SC21 (layers 5 - 7 of OSI and general architectural matters) and TC97/SC20. The chairmen of SC6, SC20 and SC21 have agreed the details of a definition of responsibilities; the effect of this is to leave the controlling responsibilities for protocols and architectural matters with SC6 and SC21 as appropriate, whilst SC20 has controlling responsibility for security matters with an emphasis on application of encipherment algorithms. We should note that a second part of the OSI architecture definition standard covering data security is in an advanced stage of preparation; this work is proceeding within SC21.

To complete this short review of present activities and trends in the data security standards area we must mention the work of TC68 (banking procedures). TC68 is producing a series of standards with emphasis on message authentication and key management. Work is proceeding on a standard describing the mechanism of message authentication (wholesale) and on a standard specifying algorithms (including the DES) that may be used for message authentication. Another important standard in preparation is that on wholesale financial key management, developed from ANSI standard X9.17. Standards for retail message authentication and key management can be expected to follow. Note, however, that there is no sign as yet of work under TC68 to use public key cryptography for banking purposes; all the known standards work uses symmetric cipher systems.

Much work remains to be done on data security standards and the abandonment of work on encipherment standards will free effort for making progress in the remaining areas.

References

1. National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standard 46, January 1977.

2. American National Standards Institute. Data Encryption Algorithm. American National Standard X3-92, 1981.

3. Desmedt, Y., Quisquater, J-J, & Davio, M. Dependence of output on input in DES; small avalanche characteristics. Proc. Crypto'84, Springer-Verlag, Lecture Notes on Computer Science 196, 1985, pp. 359-376.

4. Kaliski, B.S., Rivest, R.L., & Sherman, A.T. Is DES a pure cipher? (results of more cycling experiments on DES). Proc. Crypto'85, Springer-Verlag, Lecture Notes on Computer Science 218, 1985, pp. 212-226.

5. Shamir, A. On the security of DES. Proc. Crypto'85, Springer-Verlag, Lecture Notes on Computer Science 218, 1985, pp. 280-281.

INTEGRATING CRYPTOGRAPHY IN ISDN

Kåre Presttun
Standard Telefon og Kabelfabrik A/S
Po. Box 60, Økern, N-0508 OSLO 5

1. INTRODUCTION

Security services in ISDN have been briefly discussed earlier in the literature [25, 26 and 27]. This paper deals with the protocol aspects of integrating cryptography in ISDN.

2. AUTHENTICATION AND KEY DISTRIBUTION

Authentication and key distribution should be based on the CCITT SG VII "Authentication Framework" [1]. The framework uses public key cryptography for authentication and optionally key distribution.

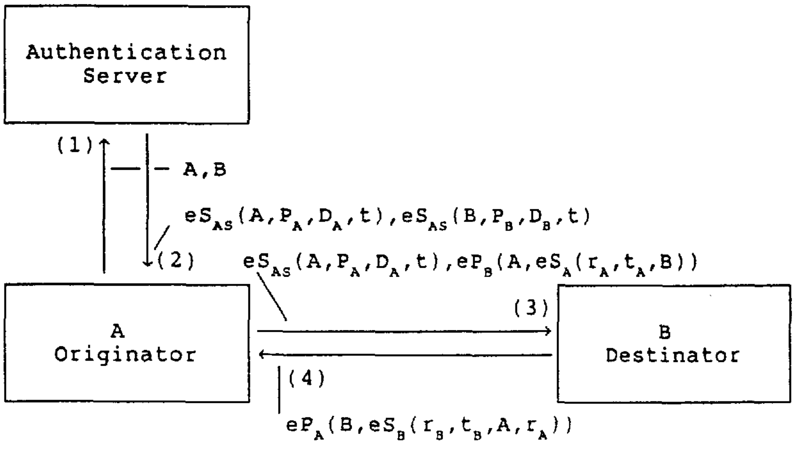

Authentication Server
(1)
A,B
$eS_{AS}(A,P_A,D_A,t),eS_{AS}(B,P_B,D_B,t)$
(2)
$eS_{AS}(A,P_A,D_A,t),eP_B(A,eS_A(r_A,t_A,B))$
(3)
A Originator
B Destinator
(4)
$eP_A(B,eS_B(r_B,t_B,A,r_A))$

Figure 1: Authentication with certificates

With this protocol A and B do not have to reveal any secret to the Authentication Server (the directory), which contains the public keys of all the users.

In the first message A says to the AS: I am A, I will talk to B. This message can optionally be signed with A's secret key. The signature can be used by the AS to see if somebody is trying to impersonate A, requesting certificates from AS. It has no security implications beceause only the real A can create message 3 anyway, but can avoid that A get billed for certificates requested by somebody else.

The reply comes back with two certificates signed with the secret key of the AS (eS_{AS}), (eS_{AS} must be interpreted as the signed massage hash). These certificates contain the public keys of A and B (P_A, P_B), and the first and last day they are valid (D_A, D_B). This is the current CCITT format. We will recommend also to include the current time (t) as recommended by ECMA [17], and shown on the figure. This means that the certificates must be generated on line, and not off line as proposed by CCITT. The reason for this is that the current recommendation does not provide adequate means for revocation of cerificates.

In the third message, A forwards his certificate to B and appends an authentication token containing a random number he has generated and his time protected by B's public key, and signed by A.

The fourth message is B authenticating himself to A by sending a random number, his time and returning A's random number in his authentication token.

If a real time communication is not available, as in electronic mail, the fourth message cannot be used, and we end up with a one way authentication scheme instead of two way. If A and B want to send encrypted data, they can use r_A as a key for encryption A to B and r_B for encryption B to A. Alternatively they can form a common key by adding r_A and r_B bit by bit modulo two. The data encryption can be performed by a conventional encryption algorithm.

Alternatively the tokens can be modified to be used with the exponential key exchange as proposed in [25]. They will then read:

$$A, eS_A(\alpha^X \bmod q, t_A, B) \tag{1}$$
$$B, eS_B(\alpha^Y \bmod q, t_B, A, t_A) \tag{2}$$

where X and Y are random numbers in the range 1...q.

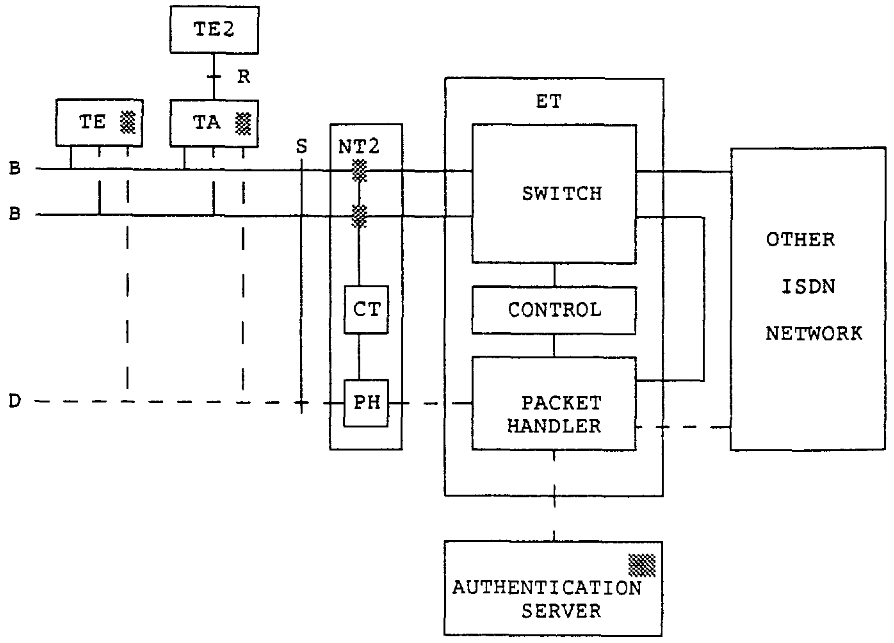
This gives us a general authentication and key distribution method for both real time communication and store and forward type communication.

3. CRYPTOGRAPHY IN ISDN

One of the main properties of ISDN (Integrated Services Digital Network, see [2,3,4]) is that a signalling/data channel (D), independent of the information channels (B), always is available to the Terminal Equipment or Terminal Adapter (Figure 2). This channel can be used for key distribution and security service management. The S interface is the standardized ISDN Basic Access, and can act as a bus with up to 8 terminals connected.

3.1 Location of the crypto processes

Proposed locations of the crypto processes are shown on figure 2. Because we want to use the D-channel for key distribution, the crypto processes must be located at points where D-channel layer 3 is processed. Possible locations are then TE, TA, NT2 and ET.



▨ Crypto Module

Figure 2: ISDN model with crypto processes located

3.2 Carrying keys

To implement the authentication framework, the first two messages (see figure 1) are transferred using the "User-to-user signalling via temporary signalling connection" facility [7]. The messages are conveyed in a new "encryption" information element in the SETUP, CONNect and USER INFOrmation messages. The "encryption" information element should carry higher layer protocols that transfer the request to the authentication server and the reply with the cerificates. This new information element should be treated like a user-user information element by the network, alternatively the user-user information element could be used. The benefit of introducing this new encryption information element is from the network operators point of view that there may be a different policy for the amount of traffic etc. to be carried in this element compared to the policy for the user-user information element.

How higher layer protocols are inserted into the "encryption" information element is shown in figure 3.

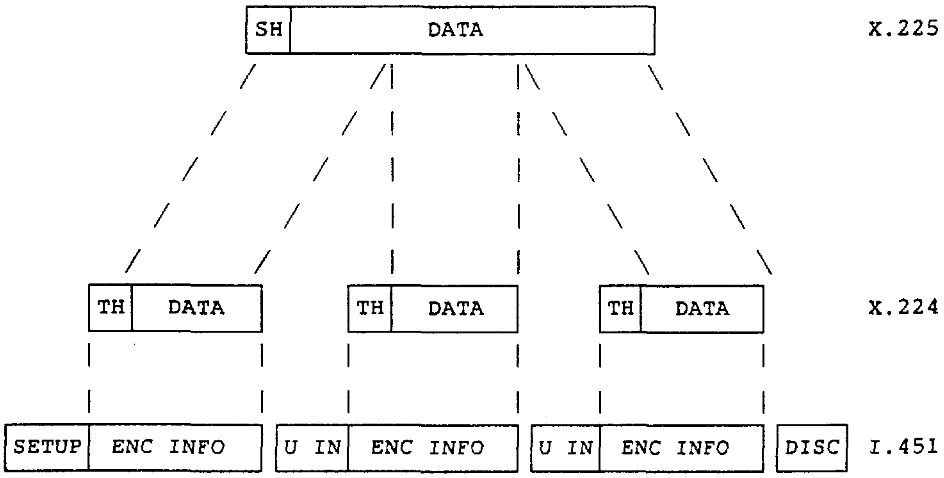


Figure 3: Inserting higher layer protocols into the D-channel protocol

The transport protocol can be of class 0 (simple class), and is needed to do segmenting and reassembling of Transport Service Data Units [15]. This is so because the length of the user-user information element is limited to 32 bytes. Embedded in the TSDU's is the Session Protocol Data Units [16]. To do a transfer of certificates and authentication tokens, only the kernel part of the session protocol is needed.

Messages 3 and 4 are transferred using the "User-to-user signalling in association with a B-channel connection" and are also conveyed in the "encryption" information element in the SETUP, CONNect and USER INFOrmation messages.

3.3 Secure Bearer services

An ISDN bearer service [5] is a service for transport of information through an ISDN network. An example of a bearer service is: 64 kbit/s, transparent, 8 kHz integrity structure, circuit switched. Our feeling is that the ISDN bearer service should not provide any end to end security measures. However, the request for bulk-encrypted trunklines could be signalled on a per call basis. This could be done by giving the encrypted trunk network a transit network code and use the "Transit network selection" information element in the SETUP message [7] to signal the request. Encryption could be provided on trunks with capacity depending on traffic requirements.

3.4 Secure Teleservices

A teleservice in ISDN [6] is a fully standardized end user service. Examples include: Telephony and Teletex. Teleservices should have security functions standardized as options at the presentation layer.

Looking at digital telephony from the OSI point of wiew, it can be regarded as having layer 1 as 64 kbit/s unrestricted with 8 kHz structure or another capability, layer 2 - 5 empty, and the voice coding method specified as layer 6 transfer syntax. The transfer syntax can be octets coded as A-law PCM or μ-law PCM, or other standardized coding methods. Thus encryption can be done at layer 6 (on the B-channel), and key management can be done via the D-channel. The signalling to indicate telephony should then be done in the "Bearer capability" information element, with information transfer capability set to unrestricted digital information, and layer and protocol (layer 1) identification set to appropriate rate and structure. Further should the actual coding and teleservice be indicated in the "High layer compatibility" information element. This is however not in line with the current understanding of bearer [5] and tele services [6], and the way a connection is requested [7]. But it seems to be in line with the basic definition of services [4] and the OSI model [14]. Here it is an inconsistency in the work done by CCITT in the previous study period.

In the recommendations for bearer services and signalling [5,7], speech transmission is regarded as a bearer service, while the actual teleservice, Telephony, is unspecified. In the signalling system the speech coding is signalled as "user information layer 1 protocol".

A proposed way to do security enhancements to Telephony in ISDN is: Signal the way now specified in the signalling system, include the "encryption" informaton element in SETUP and CONNect, and indicate encryption in the "CCITT-standardized facilities" information element in SETUP message to prevent the network from doing signal processing on the encrypted voice signal.

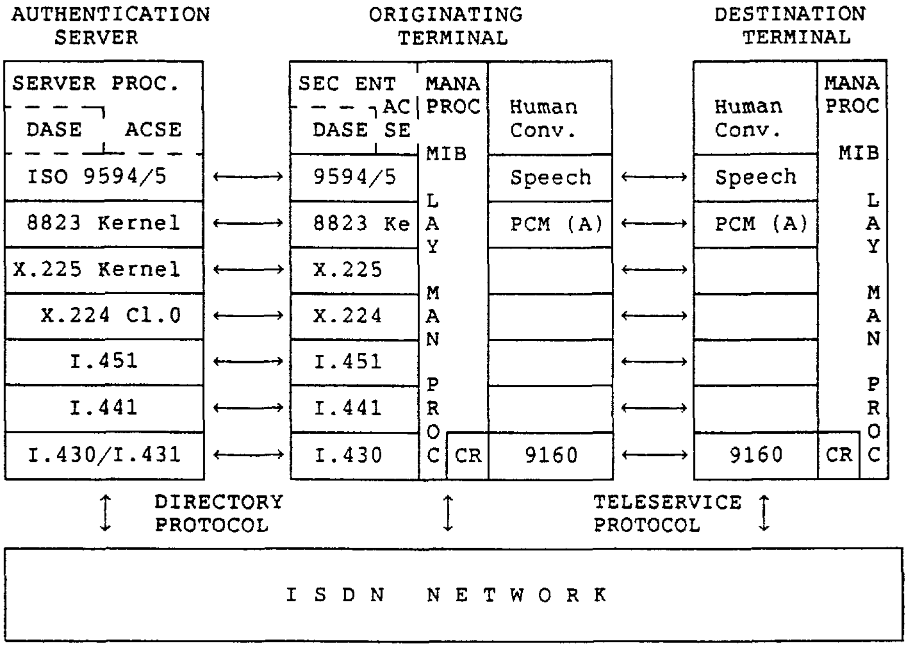


Figure 4: Protocol System for Secure Voice

Figure 4 shows how the protocol hierarchy can be built up. On top of the session protocol we find the kernel of the presentation layer [20]. As there is no need for context management, only the kernel part of the presentation protocol is needed. The Association Control Service Elements (ACSE) [19] operate on the session kernel functions and control the association for key material transfer. The Directory Access Service Elements (DASE) [23] operates on the Directory Access Protocol (DAP) [24] and is the actual application protocol used to retrieve the cerificates.

On top of the service elements is the actual server process [22]. The management process in the originating terminal communicates with the server process via these service elements. The retrieved certificates are kept in the Management Information Base (MIB) [18]. The management process communicates with cryptoservices in the terminal to verify the certificates and to generate keys and authentication tokens. The keys for data encryption are then installed in the crypto process at the actual layer by the Layer Management Process.

For voice the encryption takes place at layer 1, using appropriate parts of the physical layer encryption standard [21].

To implement security in other teleservices than telephony, the security services should be implemented at layer 6. For packet mode terminals both the association with the authentication server and the end to end association should be established as normal packet mode calls. Protocols to be used for authentication and key distribution, utilizing the authentication framework, and data transfer are shown in figure 5.

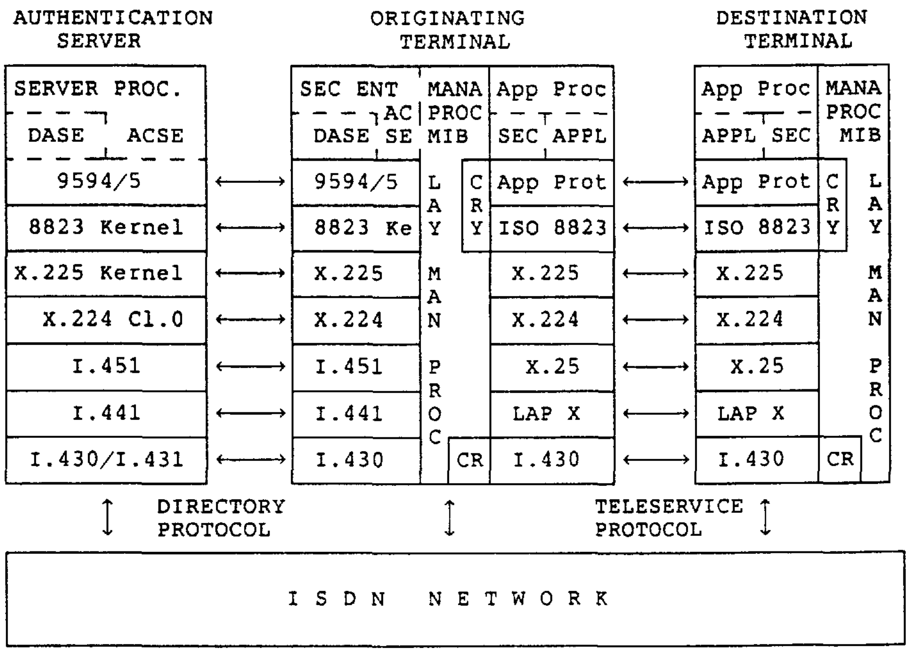


Figure 5: Protocol System for Secure Teleservices

The main difference from the voice case (in figure 4) is the protocols used for data transfer. The protocols are specified for the actual teleservice, and figure 5 shows the general case.

3.5 Support of existing terminals

A series of recommendations for the support of existing terminals on an ISDN have been developed [8,9,10,11,12]. We will recommend security enhancements to be worked out for circuit switched services only. For packet mode terminals the security services should be built into the higher layer protocols.

For circuit mode connections, a service with automatic key distribution can be developed if both terminals are connected to an ISDN. Whether it can be done with one of the terminals connected to a CSPDN (Circuit Switched Public Data Network) with maximum integration [9] needs further study. The minimum integration case does not seem to be applicable for automatic key distribution.

Encryption can be implemented, superposed on circuit switched services [9,11,12] according to ISO recommendations [21], in TAs (see figure 2) utilizing the D-channel for keymanagement.

4. ISDN AUTHENTICATION SERVER

An ISDN authentication server should be connected to a D-channel at the lowest layer in the exchange hierarchy. The data rate of the channel is dependent on the traffic. By indicating the higher layer protocols used for communication with the AS when a call to the AS is made, it is possible to build an AS common to all needs.

When the number of terminals with encryption capabilities increases, there will be a need for networks of authentication servers, constituting a distributed authentication system. At higher layers in the exchange hierarchy where the D-channel protocol is not available, ISUP [13] in signalling system No. 7 must be used for communication between the exchange and the AS. The communication between AS's will be based on the Directory System Protocol (DSP) [24]. There does not exist any direct relationship between the hierarchy of exchanges and the hierarchy of AS's as both hierarchies are developed according to traffic demands in the two systems.

5. CONCLUSION

In this paper it has been illustrated how existing work in ISO, CCITT, and ECMA can be utilized to integrate cryptography in ISDN. Necessary protocol mechanisms have been selected in such a way that interworking with cryptographic functions in other networks than ISDN should be possible.

There is a lot of standardization issues that must be addressed by CCITT and ISO before the security services can be implemented at a large scale internationally. It is expected that results will come from CCITT during the next studyperiod 1989 - 1992.

REFERENCES

[1] CCITT SG VII, Draft Recommendation X.ds7, The Directory - Authentication Framework, Geneva, October 1986.

[2] CCITT red book, I.120, Integrated Services Digital Networks (ISDNs), Malaga-Torremolinos 1984.

[3] CCITT red book, I.130, Attributes for the Characterization of Telecommunication Services Supported by an ISDN and Network Capabilities of an ISDN, Malaga-Torremolinos 1984.

[4] CCITT red book, I.210, Principles of Telecommunication Services Supported by an ISDN, Malaga-Torremolinos 1984.

[5] CCITT red book, I.211, Bearer Services Supported by an ISDN, Malga-Torremolinos 1984.

[6] CCITT red book, I.212, Teleservices supported by an ISDN, Malaga-Torremolinos 1984.

[7] CCITT red book, I.451 (Q.931), ISDN user-network interface layer 3 specification, Malaga-Torremolinos 1984.

[8] CCITT red book, I.460, Multiplexing, rate adaption and support of existing interfaces, Malaga-Torremolinos 1984.

[9] CCITT red book, I.461 (X.30), Support of X.21 and X.21 bis based DTEs by an ISDN, Malaga-Torremolinos 1984.

[10] CCITT red book, I.462 (X.31), Support of Packet Mode Terminal equipment by an ISDN, Malaga-Torremolinos 1984.

[11] CCITT red book, I.463 (V.110), Support of DTEs with V-series type interfaces by an ISDN, Malaga-Torremolinos 1984.

[12] CCITT red book, I.464, Rateadaption, multiplexing and support of existing interfaces for restricted 64 kbit/s transfer capability, Malaga-Torremolinos 1984.

[13] CCITT red book, Q.761, Functional Description of the ISDN User Part of Signalling System No. 7, Malaga-Torremolinos 1984.

[14] CCITT red book, X.200, Reference model of Open Systems Interconnection for CCITT applications, Malaga-Torremolinos 1984.

[15] CCITT red book, X.224, Transport Protocol Specification for Open Systems Interconnection for CCITT applications, Malaga-Torremolinos 1984.

[16] CCITT red book, X.225, Session Protocol Specification for Open Systems Interconnection for CCITT applications, Malaga-Torremolinos 1984.

[17] ECMA/TC32-TG9/87/12, CCITT SG VII Contribution, Security of Directories, February 1987.

[18] ISO 7498/4, Open Systems Interconnection - OSI Management Framework.

[19] ISO/DP 8649/2, Open Systems Interconnection - Association Control: Service Definition.

[20] ISO/DIS 8823, Open Systems Interconnection - Connection Oriented Presentation Protocol Specification.

[21] ISO/DP 9160, Information Processing - Data Encipherment - Physical Layer Interoperability requirements.

[22] ISO/DP 9594/1, Open Systems Interconnection - The Directory - Part 1: Overview of Concepts, Models and Secvices.

[23] ISO/DP 9594/3, Open Systems Interconnection - The Directory - Part 3: Access and System Service Definition.

[24] ISO/DP 9594/5, Open Systems Interconnection - The Directory - Part 5: Access and System Protocols Specification.

[25] B. O'Higgins, W. Diffie, L. Strawczynski, R. de Hoog, Encryption and ISDN - A Natural Fit, Proc. ISS'87, Phoenix, March 15-20, 1987, pp 863 - 869.

[26] Kåre Presttun, Security Measures in Communication Networks, Electrical Communication, Vol 60 No 1, 1986, pp 63 - 70.

[27] K. Siuda, Technische Massnahmen für die sichere Informationsübertragung in zukünftigen Fernmeldenetzen (ISDN), Bull. SEV/VSE 77(1986) 1, 11. Januar, pp 5 - 11.

NOTE

A tutorial on higher layer OSI can be found in: ISO/TC68/SC5/N174, Methodology and Guidelines for Application Protocol Development.

SPECIAL USES AND ABUSES OF THE FIAT-SHAMIR PASSPORT PROTOCOL

(extended abstract)

Yvo Desmedt [a], Claude Goutier [b] and Samy Bengio [a]

[a] Dépt. I.R.O., Université de Montréal,
C.P. 6128, succ. A, Montréal (Québec), H3C 3J7 Canada

[b] Centre de calcul, Université de Montréal,
C.P. 6128, succ. A, Montréal (Québec), H3C 3J7 Canada

Abstract

If the physical description of a person would be unique and adequately used and tested, *then* the security of the Fiat–Shamir scheme is *not* based on *zero-knowledge*. *Otherwise* some new frauds exist. The Feige–Fiat–Shamir scheme always suffers from these frauds. Using an extended notion of subliminal channels, several other *undetectable* abuses of the Fiat–Shamir protocol, *which are not possible with ordinary passports*, are discussed. This technique can be used by a terrorist sponsoring country to communicate 500 new words of secret information each time a tourist passport is verified. A non-trivial solution to avoid these subliminal channel problems is presented. The notion of *relative* zero-knowledge is introduced.

1 Introduction

Fiat and Shamir proposed (at several conferences *e.g.*, [12,11,16]) a protocol for identification which enables any user to prove his identity to any other user without shared or public keys. A variant of this protocol was proposed by Feige, Fiat and Shamir [10].

In 1986 Desmedt and Quisquater [8] already discussed a fraudulent use of the Fiat–Shamir protocol. Their remark was mainly that the Fiat–Shamir protocol identifies *secret information* instead of identifying the person. Thus some persons (*e.g.*, Alice) could *deliberately* "create" a second person (*e.g.*, using cloning) such that both can claim to be Alice. Their solution to this problem is not considered in this paper, except when appropriated.

In our paper we first explain the first version [16] of the Fiat–Shamir protocol, the more general version [12] and the Feige–Fiat–Shamir version [10] (see Section 2). We show that the Feige–Fiat–Shamir scheme suffers from *new* and *well-known old* fraudulent techniques (see Section 3). The Fiat–Shamir scheme suffers from the same problems *if* physical description would not be unique or not adequately tested, *else* its security is not based on zero-knowledge (see Section 3). In Section 4 we discuss several frauds which are in fact extensions of the subliminal channel idea of Simmons [17]. We will show that several subliminal channels can be brought in all the actual versions of the (Feige–)Fiat–Shamir schemes. Their dangers will be briefly discussed in the same section. Section 5 presents a solution to solve the subliminal channel problem in the Fiat–Shamir and Feige–Fiat–Shamir scheme.

Some of the frauds which will be discussed in this paper could be misinterpreted as being politically oriented. To avoid this, non-existing countries will be taken as example, so we will speak about αland, βland and so on.

2 The (Feige–)Fiat–Shamir protocol

The protocol is explained in the case it is used for passport purposes. Evidently it can also be used for other identification purposes such as credit cards. Several other applications were discussed by Fiat–Shamir [11].

We will first explain the basic version of the protocol [16], then the more general version [12] and finally the latest version [10].

2.1 The basic version

The protocol uses as many public keys as there are countries (more generally as there are centers who issue cards or passports). We will use the symbol (integer) n for the public key of a country (center), where $n = p \cdot q$ such that p and q are secret primes only known to the center. Remark that the authentification problem of the public key is extremely reduced since the number of countries is small.

Let us now explain the start-up of the system. There exists a standard keyless (pseudo-random) one-way function f. Let us call I the "name" of an individual (*e.g.*, Alice) who wants to receive a passport from the center. To be unique I (the "name") contains relevant information about the individual; *e.g.*, the name, address and *physical description*. For each individual the center picks a (small) j such that $m = f(I, j)$ is a quadratic residue (mod n). The center calculates the smallest $\sqrt{m}$ (mod n) and gives it to the individual. We will refer to $\sqrt{m}$ as the secret identification of the individual.

If Alice wants to identify herself to Bob then they use the following ping-pong protocol. First she tells Bob her nationality, her "name" (I) and j. So Bob knows which n to use. Bob calculates m corresponding with I and j. Then the ping-pong part starts:

Step 1 Alice chooses a *random* s (mod n) which we will further call $\sqrt{t}$ and Alice squares it (mod n) to obtain t. She sends t to Bob.

Step 2 Bob sends Alice one *random* bit e.

Step 3 Alice sends then $\alpha = \sqrt{t} * \sqrt{m^e}$.

Step 4 Bob verifies by squaring. (This is trivial, because he has to verify that $\alpha^2 = t * m^e$ (mod n) and he knows m and t, because Alice has sent that.)

Somebody else could have claimed to be Alice with a probability of 1/2. To decrease this success of frauding the protocol is repeated as many times as required for security. We will call Bob the verifier and Alice the passport holder.

Shamir claimed at Marseille [16] that the last protocol is a zero-knowledge one.

The attentive reader may have remarked that the *original Shamir protocol* is slightly *different*. Indeed Shamir claims it is better to use $\sqrt{m^{-1}}$ instead of $\sqrt{m}$ to avoid calculations of inverses modulo n [10]. We now show that this claim is exaggerated and that our version is faster.

2.2 This slightly modified version is faster than the original

In the original version Bob has to verify in Step 4 that $t = \gamma^2 * m^e \pmod{n}$, where $\gamma = \sqrt{t} * \sqrt{m^{-e}} \pmod{n}$ and γ has been sent by Alice in Step 3.

First remark that from a *mathematical point of view* the verification $\alpha^2 \equiv t * m^e$ and the verification $t \equiv \gamma^2 * m^e$ are *identical*, because $\alpha \equiv m^e * \gamma \pmod{n}$ and because $\gcd(m, n) = 1$. *However* the two verifications differ from a *computational point of view*. Indeed to do this verification step (in Shamir's original version) Bob has to *wait* until he received γ from Alice, *then* he has to square it and multiply it with m (when $e = 1$). In our version Bob can multiply t and m (if $e = 1$) *while* he is *waiting that Alice calculates and sends* α, *then* when he receives α he has only to square it to verify that $\alpha^2 = (t * m^e)$, where the right-hand side was calculated *before* (as just explained).

So no inverses have to be calculated in the protocol in any of the two cases. In the two cases a squaring operation has to be done and, if $e = 1$, an additional multiplication. *However* our version is faster because a part of the calculation can be done in *parallel* (while Alice is calculating and sending).

The same remark about speed is also valid for the Fiat–Shamir [12] and the Feige–Fiat–Shamir [10] protocol. *So from now on, when we explain or use the other versions, we will use the faster adaptation.*

2.3 The Fiat–Shamir protocol

The differences are small (except some proofs) between the first version (discussed in Section 2.1) and the more general Fiat–Shamir protocol. Instead that *only one* j exists, k such j^i exists, so that k m_i $(1 \leq i \leq k)$ and k $\sqrt{m_i}$ exist. In the ping-pong protocol Bob sends k e_i in Step 2. In Step 3 now Alice sends

$$\alpha = \sqrt{t} * \prod_{e_i=1} \sqrt{m_i} \pmod{n}.$$

Bob verifies (in Step 4 by squaring, this means calculates first (in parallel with Step 3) $\beta = t * \prod_{e_i=1} m_i$ and when he receives α, he squares it to verify that $\alpha^2 = \beta \pmod{n}$.

2.4 The Feige–Fiat–Shamir scheme

By reading [10] and comparing it with [12] the differences between the Fiat–Shamir and Feige–Fiat–Shamir protocol seem small, *however* they are important. The fact that the prover does not reveal that a number is, or is not, a quadratic residue, is *not* extremely *important* in the *context of our paper*. The important difference is that in their new scheme [10, pp. 214-215] *the role of the center is enormously reduced*. We now explain this, by emphasizing this last fact.

The *only* role of the center is to publish an n of the appropriate form (the product of two large primes each of the form $4r + 3$). *And then* the center *closes*. Each individual chooses k

random numbers $S_i \pmod n$. He then chooses each $m_i = \pm S_i^2 \pmod n$, where the sign is decided randomly (and independently). He keeps the S_i secret and makes the m_i public.

Remark at this point the enormous *difference with previous protocols.* In previous protocols the *center* was calculating *square roots*, while here the *individual* calculates *squares*. *This remark will play a very important role in Section 3.2.*

The ping-pong protocol to prove Alice's identity is very similar to the Fiat–Shamir protocol. Details are not important in this context, for them see [10].

3 Physical description and related problems and frauds

In this section we first show that *if* the physical description of a person is unique and adequately used and tested, *then* the security of the Fiat–Shamir scheme is *not* based on *zero-knowledge*. Otherwise some new (and also old) frauds exist, which can severly affect the security of the identification.

3.1 Security aspects of Fiat–Shamir not related to zero-knowledge

In this subsection we assume that the physical description is *unique* and is used as a part of I (see Section 2.1), which is an input of the one-way function f. We also assume that each time that Alice's passport is verified, her I is adequately tested. This means that the verification of the physical description *is always done by the verifier* and with a 100% accuracy. In that case it is trivial to understand that the security of the Fiat–Shamir protocol is not based on zero-knowledge. Indeed there is no danger for Alice to reveal her $\sqrt{m_i}$ each time she wants to prove her identity. The verification of the identity would then be that Alice gives her I, that her physical description is tested and verified and then she reveals her j_i and her $\sqrt{m_i}$ to the verifier. There is no danger that the verifier or friends of the verifier or who so ever (else than Alice) can afterwards (or at the same moment) claim to be Alice. Indeed when they would try to use it, their physical description will be tested giving another I', giving another m', so $\sqrt{m}$ (of Alice) is not useful to them. To demonstrate it completely, suppose that they could use $\sqrt{m}$, then they could forge fake individuals! This last fraud is impossible in the Fiat–Shamir scheme due to the one-way function and square-root operation.

The above reasoning is nothing else than applying the ideas from Simmons [19] to the Fiat–Shamir protocol. Readers who are more interested in a comparison between the two systems (Fiat–Shamir and Simmons [19]) are referred to [1].

The conclusion of the above is that the ping-pong protocol is simply a waste of time and effort. The main security aspect is the fact that the physical description is unique and adequately used and tested. To better understand its importance we now discuss what happens with the Fiat–Shamir and Feige–Fiat–Shamir protocol [10] when physical description is not unique.

3.2 Major frauds possible in the Feige–Fiat–Shamir protocol

From now on we assume that the physical description of the individual is not used in the identification system or is not unique or is not adequately checked (*e.g.*, a fraud with the physical description is possible). We remark immediately that the *Feige–Fiat–Shamir* protocol [10, pp. 214–215] does *not use the physical description.* Trying to do this is nevertheless difficult, due to the fact that an individual can not calculate square roots of random numbers.

Before, in 1986, Desmedt and Quisquater [8] already discussed a fraud in the case that physical description is not unique or adequately used and checked. In such case at least three other frauds or problems are possible with passports. *Some of these frauds and problems are well-known, due to the fact that some of them are also possible with ordinary passports. However they were never cited before in the context of the security of the Feige–Fiat–Shamir protocol. We also now discuss frauds which are not possible with ordinary passports.*

Before discussing these three frauds and problems we mention that more frauds are discussed in [1]. Some of the problems we will discuss now can be solved. Discussing these solutions is out of the scope of this paper, which focus on the abuses of the protocols. Readers interested in the solutions can find them in [1] and [5].

3.2.1 Individuals having several identities

In the Feige–Fiat–Shamir protocol the center vanishes after publishing the n. The work to publish the m_i is left entirely to each individual . A clever individual can however make several entries into the public file (repeating several times the process of making new m_i) to have more than one name (identity) at the same moment. This trick is very useful for persons who wants to fraud with taxes. Another application of this fraud is that it allows you to commit a crime and disappear. Hereto you first publish several identities. One of them you never use. Then later, you identify yourself with the one you never use and commits the crime immediately so that the one who verified your identity is a witness. A search starts to find you back, however you have stopped to use that identity!

Consequently each individual must have *only one identity*. (If pseudonyms are used (as in [4]), then one individual may only go under the name of one pseudonym in one organization.) So we conclude that an organization (which we call the *center*) has to exist which verifies that one individual has only one identity! So the center has to manage the public file of the m_i. The center has to be trusted that an individual will not receive two identities. *However* if physical descriptions are not unique or not tested or similar problems exist, *the center itself is not able to recognize an individual when he applies a second time for an identity.*

Two solutions exist such that the center can guarantee the above uniqueness. The first one uses the terrifying idea of tamperfree babies which are uncloneable and containing a *unique* number (or name) as a part of their genetic code [1]. So this solution makes individuals testable unique. The other solution is not water-tight. In the last solution each individual can only apply for an identity when he is born. In fact his parents have to do this (or those who care for him). In several countries no fingerprints are taken from the newborn, so parents could ask the doctor for two birth-certificates and so apply for two identities. So if physical description is not tested or not unique (or similar), then the uniqueness of an individual in the end is based on trust.

We here remark that the uniqueness aspect of an individual is not a part of the definition of identity in the Fiat–Shamir [12] and in the Feige–Fiat–Shamir sense [10]. So the above fraud does not break the Feige–Fiat–Shamir protocol from a strictly mathematical point of view. *We* *also remark that some of the above aspects are well-known to officials dealing with identification as immigration departments, counselors and so on.*

3.2.2 The mafia fraud

We call this fraud the mafia fraud as a consequence of Gleick's article [13] quoting Shamir (related to the protection of credit cards with his [12] protocol): "I can go to a Mafia-owned store a million successive times and they still will not be able to misrepresent themselves as me".

Let us now explain the fraud. A identifies himself to B. The latter is collaborating with C and C impersonates A and tries to claim to be A. Then, D checks the identity of C who is claiming to be A. To make it easier to understand, B is the owner of a mafia-owned restaurant, C is a member of the same mafia-gang and D is a jeweller. A and D are not aware of the following fraud. At the moment that A is ready to pay and ready to prove his identity to B, B informs C that the fraud is starting. This is done by using a secret radio-link between C and B. The identification card of C communicates also, using such a radio-link, with the equipment of B. At this point, C makes his choice of the diamond he wants to buy and so D is starting to check "C's" (in fact A's) identity. While D is checking the identity, C and B's role is only to sit in the middle between A and D. So B and C pass all questions and all answers related to the mathematical part of the identification going from D to A and vice-versa. So even if D is aware that an identification procedure over the telephone could not work, another person could come physically to his store and D would not be aware that he is remotely checking A's identity. Evidently this fraud does not work in all circumstances *e.g.*, when the verification of C's claimed identity by D cannot be synchronized with the verification of A's identity by B. In our example the fraud is facilitated because A goes frequently to mafia-owned stores. Nevertheless Shamir claims [13] that there is no danger.

A similar fraud is possible when A *is willing to collaborate* with C immediately! For more details see [1].

It is important to remark here that the above fraud is a real-time fraud. This does not exclude its generality, related to the Feige–Fiat–Shamir protocol because that protocol only works in real-time. Indeed they mentioned in [10, p. 214] that identification is a real-time operation!

3.2.3 Renting passports

The fraud we discuss now is also possible with ordinary passports. However it was never cited in the context of the "secure" Feige–Fiat–Shamir scheme.

We now explain why a user is sometimes willing to hire out his passport. Suppose that the identification system is used for passport purposes. Brigitte is not able to receive a visa to travel to αland. However she has a good reason to travel to αland. She is rich, but does not want to bribe the visa-office, because she does not like to start her trip with a lot of trouble. Alice proposes to hire her passport to Brigitte. In other words, Alice simply tells Brigitte her secret identification $\sqrt{m}$. (Remark that this fraud will not be detected by αland.) The advantage for Alice is not only money, but now she can commit a crime with a perfect alibi. Hereto Alice commits a crime while Brigitte is travelling in αland (pretending to be Alice). Alice has evidently a perfect alibi, because Alice (in fact Brigitte) has proven several times (at the moment of the crime) that she was in αland. Remark that this last fraud is very close to Russian roulette. Indeed Brigitte could also have committed a crime while she is renting Alice's identification system and be certain she will not be arrested. She commits this crime in the neighbourhood of Alice's home. Brigitte first identifies herself as being Alice (which is possible), then commits the crime (*e.g.*, in the same room) and runs away! Alice can evidently pretend that she hired her secret to Brigitte, but who will believe her!

One could conclude that the identification protocol is in fact a protocol of identification of the secret $\sqrt{m}$ instead of identifying the individual. This is the same conclusion as in [8]. To overcome this problem Desmedt and Quisquater proposed a technique that prevents copying of $\sqrt{m}$. Remark that their solution does however not solve this fraud!

Because the above fraud is also possible with the actual passports (*e.g.*, by tampering with the photo) it seems to be inherent to passports in general. This remark remains valid even if biologic information related to the individual is a part of the I as a consequence of cloning (see also [8]).

The tamperproof babies solution (see Section 3.2.1 and [1]) solves the above problem. The only practical solution that the authors see is that each individual is forced to show his passport very frequently, (*e.g.*, each day, at each corner of the street) as in a police state. Then the above fraud can be extremely reduced.

An important conclusion here is that some frauds are possible if the user allows somebody to fraud. One could compare this with the first ideas about signatures. After that Diffie and Hellman [9] proposed the idea of signature, Saltzer [15] said that its security was limited when the undersigned claimed that his secret key was stolen, while it was in fact deliberately made public. The idea of the fraud with passports is similar. The above remarks *generalize* to all cases wherein somebody considers *loosing* (partially) *his identification secret* as of *minor importance*.

In the following sections frauds *which are not possible with ordinary passports*, are discussed. In the new frauds one does not necessarily loose his identification secret.

4 Traffic-analysis-free communication using the (Feige–) Fiat–Shamir protocol

4.1 An extension of the subliminal channel idea

Simmons [17] presented the idea of a subliminal channel as a part of a channel with authentication facilities. Simmons assumed that two prisoners are allowed to send *messages* in full view of the warden such that the messages are completely open (and presumably innocuous) to the warden. The last one however agrees that the prisoners may authenticate the *communication*. Simmons explains how a subliminal channel can be set up [17].

The idea of Simmons can trivially be extended to the identification protocol of Fiat–Shamir. The main idea is that instead of choosing $\sqrt{t}$ and/or e randomly, *they are a part of a secret message*.

A *further generalization* of the idea of subliminal channel is that the subliminal channel is not used by the sender to communicate with the addressee. The sender uses it to communicate with an eavesdropper.

Finally we introduce the notion of *subliminal protocol.* When a protocol is used in full view of a warden, it is possible that it "hides" another protocol. In the best case (or worst case, depending of the point of view) the warden is ***not able to detect that such a subliminal protocol is used.*** The word "hided" has to be interpreted in its widest sense. An example of such a subliminal protocol will be used in Section 4.2.

It is very important to remark that the above generalizations are quite different from the original idea of Simmons. This is very clear from the viewpoint of *traffic-analysis.* Indeed if one looks at the use of "normal" (actual) passports, it happens frequently that one does ***not speak*** during its verification. In some countries when you arrive at the border you give your passport, it is verified and the agent does not even tell you that you are allowed to walk in. He simply looks to the passport of the next person. So here *nothing is communicated* except the specific message: "I am Alice". So traffic-analysis of such a message is worthless, while in Simmons idea the warden knows that a message is sent (the one the warden is able to read). A similar remark is true for our second generalization. The traffic analyst does not know who is eavesdropping. So a traffic-analyst will not be able to tell if a communication between passport verifier (or the passive eavesdropper) and passport holder (or the passive eavesdropper) is under way!

From now on we assume that all countries agreed to use the Fiat–Shamir protocol for passport purposes, instead of the paper or plastic document. All countries signed an agreement about it. We also assume that its use is yet very general. Several applications of this subliminal channel idea to passports , to credit cards and so on are discussed in Section 4.4. We will now discuss how to use the subliminal channel idea in the basic, the general Fiat–Shamir and the Feige–Fiat–Shamir protocol. There are mainly two cases: the verifier or the passport holder wants to send a message using a subliminal channel. Setting-up a *secure* subliminal channel is much more complicated in the last case than in the first one. Remark that the center can always communicate a very little bit, by choosing a special j (not applicable to the Feige–Fiat–Shamir scheme). j in fact can contain a few bits of information. We will no further discuss this last case, because it is not a real communication channel.

Let us now explain how the verifier Bob can communicate in a subliminal way to the passport holder or to an eavesdropper. Hereto he does not choose the collection of e's randomly but he lets them correspond to the encrypted message, using a secret key system or a public key system. So $\mathbf{e} = (e_1, e_2, \ldots, e_l)$ corresponds with $E_k(M)$, where M is the message. Remark that if the message is intended for an eavesdropper, then Alice (the passport holder) is not able to predict the e's. So Bob is still verifying Alice's identity with the same accuracy!

As already mentioned the set-up of a secure subliminal channel in which the passport holder communicates is much more difficult. First of all if the verifier is willing to cheat enormously there is no problem to set up a subliminal channel. The cheating consists in dropping the verification step (Step 4) in the Fiat–Shamir (or Feige–Fiat–Shamir) protocol. Indeed suppose that there is no warden who controls the protocol, the following cheating is possible. The passport holder sends the encrypted message as t. The verifier sends e. Of course, sometimes the passport holder is not able to give the correct answer. When this happens the passport holder simply sends a random number (mod n). The verifier willing to cheat, does simply not verify that the passport holder did not answer correctly.

From now on we assume that there is a warden who reads all what verifier and passport holder send. He performs himself the verification step. We could also easily assume that all this is done in public. So that everybody knows what t, e and so on are. A warden was also used by Simmons [17].

In the case that the passport holder communicates, there are *several dangers* that the passport holder *wants certainly to avoid.* Indeed he is willing to communicate with the passport verifier, but he prefers (if possible) not to loose his identification secret. He also wants that the center who issued the passport is not able to detect that a subliminal channel is used! He does not want that others (*e.g.*, the warden) detect what is going on. *General* solutions will be proposed in which we assume that the receiver of the subliminal channel does not collaborate with others to help them in detecting that a subliminal channel was used. This seems a very reasonable assumption. Also in Simmons case [17] the same assumption is valid without being mentioned by Simmons, similarly in [18].

In [6] the *general solutions will be proven to be secure against detection of their use and against revealing (a part) of the identification secret* by using reasonable assumptions, *such as* the assumption of no collaboration by the receiver (or by the sender) to help others to detect the subliminal channel or to prove that it has been used, the assumption that factorization is hard and that it is infeasible to detect if a number q (for which the Jacobi symbol $(q \mid n) = 1$) is a quadratic residue (mod n) (without the collaboration of the center or others who knows the factorization of n). *The encryption system used in the subliminal channels is a secure probabilistic public key system (e.g., the Blum–Goldwasser one based on RSA [2, pp. 298]) or a secure conventional system (e.g., based on a good pseudo-random generator, e.g. [3], used in a (synchronous) stream cipher).*

There exist several cases related to the introduction of a subliminal protocol (in which the passport holder communicates) in the Feige–Fiat–Shamir scheme. Hereto we will first discuss in Section 4.2 how to introduce it in the basic Fiat–Shamir scheme. Then we discuss how two different subliminal protocols can be introduced in the general Fiat–Shamir scheme (see Section 4.3). We will not discuss in detail how to introduce one in the Feige–Fiat–Shamir scheme. The reader can easely figure out that the same techniques as in Section 4.3 are applicable.

4.2 Introducing the subliminal protocol in the basic Fiat–Shamir

In our subliminal protocols Alice wants to send a message (using the subliminal channel). She is the passport holder. The receiver of the subliminal channel is Daisy. The verifier of the passport is Bob. It makes no difference if Bob and Daisy would be the same person, *so no collaboration of the verifier is required*, *e.g.*, by choosing the e in a special way. Daisy could for example be eavesdropping.

Let us now describe the subliminal protocol which is used during the basic Fiat–Shamir protocol. Alice wants to send the message M to Daisy. Every user, so also Alice use a known number y such that the Jacobi symbol $(y \mid n) = -1$. (Such y can easily be generated in random polynomial time.) This y may even be standard.

Step 1 Alice decides not to choose $\sqrt{t}$ randomly, but to do the following. Alice authenticates M and encrypts it to obtain C. She selects a random bit $v \in \{0, 1\}$. She selects a random number (mod n). We will call this random number $\sqrt{r}$. She squares it (mod n) and multiplies it with C and with y^v to obtain $Cry^v \pmod{n}$ and uses it as $\sqrt{t}$ for the basic Fiat–Shamir protocol, this means $\sqrt{t} = Cry^v \pmod{n}$. She now follows the basic Fiat–Shamir protocol, this means she squares $\sqrt{t}$ and sends it to Bob. (In other words she sends $C^2r^2y^{2v} \pmod{n}$ to Bob.)

Step 2 Bob chooses a (really) random e and sends it to Alice (as in the basic Fiat–Shamir protocol).

Step 3 Alice follows the basic Fiat–Shamir protocol, so if $e = 1$ she sends $\sqrt{t * m} \equiv Cry^v\sqrt{m}$ (mod n), else $Cry^v \pmod{n}$. (She is able to do this.)

Step 4 Bob verifies (as in the basic Fiat–Shamir protocol).

Step 5 When e was one, Alice restarts her subliminal protocol, so she restarts at Step 1. (This means she reencrypts M, which will give a *different* C and chooses again randomly the $\sqrt{r}$ and a random v.) Else (when e was zero) she continues the basic Fiat–Shamir protocol with $\sqrt{t} = \sqrt{r}$. So she sends r.

Step 6 Bob sends to Alice one random bit e (a new one).

Step 7 Alice follows the basic Fiat–Shamir protocol and answers what was asked. (She is able to do that because she knows $\sqrt{m}$ and $\sqrt{r}$.)

Step 8 Bob verifies.

In the case that $e = 0$ in Step 2, Daisy will later be able to receive the message M (if the ping-pong protocol was not halted at that stage (after Step 4)). From now on we assume that we are in the case $e = 0$ (in Step 2) and that the basic Fiat–Shamir protocol was not halted after Step 4. It is important to remark that Daisy also knows that e was 0 and so knows in which case she is. Thus Daisy knows that Alice has sent to Bob Cry^v as answer in Step 3. Alice sends then r in Step 5. So Daisy is now able to try to calculate C and consequently she is able to find M. Hereto she tries first $v = 0$, if she finds garbage for M then she tries $v = 1$, if she still finds garbage then Alice did not use (at this stage) the subliminal protocol. It is clear that the signature (or authentication) by Alice is necessary.

In [6] the security of the subliminal protocol is proven, after having introduced the term *relative zero-knowledge*. Loosely speaking relative zero-knowledge means that a protocol can be zero-knowledge relative to Charles but not relative to Edward. The following theorems guarantee the security.

Theorem 1 *It is infeasible for others than the receiver (and evidently the sender) of the subliminal channel to detect the use of the subliminal protocol. This is especially true for the center who issued the passport.*

Proof. See [6]. □

So for others it is infeasible to detect if the subliminal channel is used or not. The following theorem tells that there is no danger for Alice that she might reveal a part of her identification secret by using the subliminal channel. Hereto it is sufficient to prove that our *subliminal protocol* is zero-knowledge *relative to the sender Alice.*

Theorem 2 *If the assumptions used for setting up our subliminal protocol are valid, then our subliminal protocol is zero-knowledge relative to Alice.*

Proof. See [6]. □

Let us make a technical remark which is not of practical importance. If there is a collaboration between the center and Alice, then the protocol is not zero-knowledge *relative* to *the entity* formed by Alice and the center. In practice this collaboration is excluded, as we did in our assumptions.

4.3 Introducing the subliminal protocol in the general Fiat–Shamir

If one would try to adapt the above subliminal channel (see Section 4.2) trivially to the *general* Fiat–Shamir protocol, the speed would decrease exponentially when k increases. Indeed one could adapt it such that the subliminal part starts only if *all* the k e_i would be zero (see Step 5). This subliminal channel idea has to be rejected because it is completely impractical. So better solutions will be presented.

Two cases now mainly exist. In the first one the passport holder is willing to discuss a secret protocol with the verifier so that the verifier cheats by choosing his e_i such that his subliminal channel becomes easier to build! So the e_i are the output of a secure pseudo-random generator and both, passport holder and verifier know the secret key and seed. This first case is discussed in Section 4.3.1. The second solution can be used if the receiver of the subliminal channel is the eavesdropper and not the verifier. In this case the verifier will probably not collaborate by selecting the e_i in a special way! This second case is discussed in Section 4.3.2.

4.3.1 The passport holder sending to the verifier

Let us explain the subliminal protocol which is used during the general Fiat–Shamir protocol. Alice (the passport holder) wants to send the message M to Bob (verifier). Every user, so also Alice use a known number y such that the Jacobi symbol $(y \mid n) = -1$. (Such y can easily be generated in random polynomial time.) This y may even be standard.

Step 1 Alice decides not to choose $\sqrt{t}$ randomly, but to do the following. Alice calculates the $e_1, e_2, \ldots, e_k$ that Bob will send her in Step 2 (using the common pseudo-random generator). Alice authenticates the message M and encrypts it to obtain C. She selects a random bit $v \in \{0,1\}$. She selects a random number $\pmod n$. We will call this random number $\sqrt{r}$. She squares it $\pmod n$ and multiplies it with C, with y^v and with $\prod_{e_i=1} \sqrt{m_i}$ to obtain $Cry^v \prod_{e_i=1} \sqrt{m_i}$ and uses it as $\sqrt{t}$ for the general Fiat–Shamir protocol. She now follows the general Fiat–Shamir protocol, this means she squares $\sqrt{t}$ and sends it to Bob. (In other words she sends $C^2r^2y^{2v} \prod_{e_i=1} m_i \pmod n$ to Bob.)

Step 2 Bob calculates $e_1, e_2, \ldots, e_k$ similarly as Alice and sends them to Alice.

Step 3 Alice answers as in the general Fiat–Shamir protocol, *even if* the e_i that she just has calculated differ from the one that Bob just has sent.

Step 4 Bob verifies (as in the general Fiat–Shamir protocol).

Step 5 When the e_i that Alice has calculated in Step 1 do *not* correspond with the e_i that Bob has sent in Step 2, *then* Alice restarts her subliminal protocol, so she restarts at Step 1. This means she reencrypts M, which will give a *different* C *from before* and chooses again randomly the $\sqrt{r}$ (and so on). Else (when e_i match) she continues the general Fiat–Shamir protocol with $\sqrt{t} = \sqrt{r}$. So she sends r.

Step 6 Bob sends to Alice random bits, e_i.

Step 7 Alice follows the general Fiat–Shamir protocol and answers what was asked. (She is able to do that because she knows $\sqrt{m_i}$ and $\sqrt{r}$.)

Step 8 Bob verifies.

In the case that the e_i matched, Bob is able to receive the message M. Indeed in that case Alice answers (in Step 3) $Cry^v \prod_{e_i=1} m_i$. Because Bob knows also the m_i and because he received the r in Step 5, he is able to figure out Cy^v.

The proof of the security of the last subliminal channel is similar to the one in Section 4.2 (for details see [6]).

If Alice *really trusts* Bob, *then* she can double the capacity of her subliminal channel. Hereto it is sufficient that she sends $C^2 \prod_{e_i=1} m_i$ in Step 1 (where e_i is the output of the pseudo-random generator). She can then drop Step 5–8. *However if Bob does not send the same e_i in Step 2*, then *Alice gets into real trouble.* Indeed then she has the choice: either to answer, but then she looses some secret about her $\sqrt{m_i}$; or either she can refuse to answer or sending a wrong answer. It is clear that she will have trouble with Bob or with the warden (who tries to catch subliminal channel users). Remark that it can also be a simple misunderstanding due to lack of synchronization of the pseudo-random generators.

We here finally remark that the above subliminal channel could also have been used in the case of the basic Fiat–Shamir scheme, when the passport holder communicates with the verifier. Nevertheless that this last idea is more optimal, the authors didn't want to make the reading of Section 4.2 too complicated by splitting it into two cases.

4.3.2 The passport holder communicating with the eavesdropper

The startup and conditions are similar as in Section 4.3.1, *except* that the encrypted message M is intended for Daisy (eavesdropper), and except that the *center of the passport holder has to collaborate partially.* This collaboration exists in choosing n as the product of two primes and the center has to publish a number z which is a quadratic non-residue with Jacobi symbol $(z \mid n) = 1$ (-1 is such a number if n is the product of two primes each of the form $4r + 3$). If the center refuses to do this, then the following subliminal channel can still be used (hereto replace z by 1), but the center can detect its use. The subliminal protocol has also to be adapted, because Bob (the verifier) is not aware of the use of the subliminal channel by Alice and Daisy. The authors found their inspiration for the following subliminal protocol from the use of GF(2^m) in several factoring algorithms.

Step 1 Alice decides not to choose $\sqrt{t}$ randomly, but to do the following. Alice authenticates M and encrypts it to obtain C. She selects a random bit $v_1 \in \{0,1\}$. She selects a random number (mod n), which we will call $\sqrt{r_1}$. She squares it (mod n) and multiplies it with C and with y^{v_1} to obtain $Cr_1y^{v_1} \pmod{n}$ and uses it as $\sqrt{t}$ for the basic Fiat–Shamir protocol, this means $\sqrt{t} = Cr_1y^{v_1} \pmod{n}$. She now follows the general Fiat–Shamir protocol, this means she squares $\sqrt{t}$ and sends it to Bob.

Step 2 Bob chooses a (really) random $\mathbf{e}^1 = (e_1^1, \dots, e_k^1)$ and sends it to Alice (as in the general Fiat–Shamir protocol).

Step 3 Alice follows the general Fiat–Shamir protocol, so answers. Alice and Daisy each store the binary vector $\mathbf{e}^1$. They each set the number $w = 1$.

Step 4 Bob verifies (as in the general Fiat–Shamir protocol).

Step 5 *If* $\mathbf{e}^1$ can be written as $\mathbf{e}^1 = \mathbf{0}$ *or* as $\mathbf{e}^1 \equiv b_2\mathbf{e}^2 + b_3\mathbf{e}^3 + \cdots + b_w\mathbf{e}^w \pmod{2}$ (where b_j are binary), *then* go to Step 9, *else* the following happens. Alice and Daisy increase w by 1. Alice selects random bits v_w and u_w and a random number (which we call) $\sqrt{r_w}$ (mod n). Alice then sends random bits $r_w^2 y^{2v_w} z^{2u_w} \pmod{n}$ to Bob.

Step 6 Bob chooses a (really) random $\mathbf{e}^w$.

Step 7 Alice follows the general Fiat–Shamir protocol, so answers. Alice and Daisy each store the binary vector $\mathbf{e}^w$.

Step 8 Bob verifies as in the general Fiat–Shamir protocol. *Jump back to Step 5.*

Step 9 Alice sends to Bob $r_1 \prod_{b_j=1} r_j \pmod{n}$. So now Daisy is able to calculate C and M. Indeed Alice has sent $\alpha = Cr_1y^{v_1} \prod \sqrt{m_i^{e_i^1}}$ in Step 3 and has sent $\beta_j = Cr_jy^{v_j}z^{u_j} \prod_i \sqrt{m_i^{e_i^j}}$ in Step*s* 7. Daisy can now make the product $\alpha \prod_{b_j=1} \beta_j \equiv C\gamma\eta r_1 \prod_{b_j=1} r_j \pmod{n}$ where $\eta \equiv y^{v_1} \prod_{b_j=1} y^{v_j} z^{u_j} \pmod{n}$

$$\gamma \equiv \prod_i \sqrt{m_i^{e_i^1 + \sum_{j=2}^{w} b_j e_i^j}} \pmod{n}.$$

Because $e_i^1 + \sum_{j=2}^{w} b_j e_i^j \equiv 0 \pmod{2}$, only powers of m_i are in γ and no powers of $\sqrt{m_i}$, so Daisy can calculate γ herself and find C. Hereto Daisy has to eavesdrop what Alice has just sent ($r_1 \prod_{b_j=1} r_j$) and has to figure out what η is. So Daisy has to try all possible v_1, v_2, ..., v_w and all u_2, ..., u_w, so maximum k^2 trials.

Step 10 The protocol continues as before. This means that $r_1 \prod_{b_j=1} r_j$ is treated as t. So Bob chooses the next e_i and Alice answers and Bob verifies.

The reader wondering about the problem that Bob (the verifier) can decide to stop in a stage that is not the end of the subliminal protocol, has to take the following into consideration. If Daisy is able to intercept the next verification of Alice's passport (by who so ever) then no problem exists.

Proofs of security can be found in [6].

This subliminal protocol can also be used when the passport holder wants to communicate with the *verifier* knowing the verifier's public key, but having no secret key in common. Although, this last problem can sometimes trivially be avoided. Indeed the Fiat–Shamir scheme does not exclude that the passport verifier sends his j_i under some permutation. If $k \geq 40$ then the number of possible permutations is large enough to send (under encrypted form, using the verifier's public key) the actual secret DES key and seed which will be used. This key and seed could then be used to make the pseudo-random which was required in the subliminal protocol of Section 4.3.1.

4.4 Applications

We now discuss applications of our subliminal channels. One example is discussed separately in Section 5.1.3. It is clear that there are extremely dangerous uses of the discussed subliminal channel. It is not the purpose of this paper to help criminals, so the authors restrict themselves. We discuss briefly some examples.

The use by criminals to communicate secretly is a major danger when the Fiat–Shamir protocol would be adapted for passports without modifications. It can also be used by the godfather to communicate in a tamper-free way with the mafia members. Criminals infiltrated in the police can use it to communicate with their gang, to avoid for example that the gang is caught up.

If citizens of βland are allowed to choose their random themselves, they can ask politic asylum in αland, even if the secret police of βland watches them extremely closely. Hereto it is sufficient to use one of the subliminal protocols using the public key of αland, specially published for this purpose. If banks use the Fiat–Shamir protocol then bank-clerks could use it to slip information about the customs of the bank, to a company or gang or state in an undetectable way. Suppose now that a machine is used for checking the identity of the customer of the bank. The programmer of the machine can however use the subliminal channel to leak information in one or the other direction. Industrial espionage could easily be done using this method. A visitor visiting a company has to check in. The one who verifies the visitor's identity can use the subliminal channel. People forced by the secret police of βland to fly out αland against their will can also alert αland using the subliminal channel and the public key of αland.

By combining the ideas that the verifier can communicate with the passport holder *and* vice-versa, another interaction can be hidden inside the verification interaction. In extreme one can have a zero-knowledge protocol hidden inside another zero-knowledge protocol. For more details see [6].

4.5 Efficiency

Let us define the efficiency of a subliminal channel as the rate of the subliminal channel divided by the rate of the main channel. In [6] is proven that the efficiency of the subliminal channels discussed in Section 4.2, 4.3.1 and 4.3.2 are respectively: $1/2*1/3$, $1/2*1/2$, $1/2*1/(k+2)$. The first 1/2 always originates from the fact that if the subliminal message is sent hidden in t, then Alice can no longer send new subliminal messages when she has to answer, otherwise the verifier and/or warden will detect that.

5 How to avoid subliminal channels into the (Feige–)Fiat–Shamir protocol

5.1 An attempt

An anonymous reader of a preliminary version [7] of this paper proposed the following solution to overcome the subliminal channel:

> On the subliminal channels, the authors should consider the following. A truly random source will be built into the hardware of a Fiat–Shamir device. This random source would have to be circumvented during execution of the subliminal channel. This modification would be detected by physical inspection at the time.

There are however several ***problems*** related to this solution (similar to the one already mentioned in [7, p. 14]). The main ones are that

1. the above solution can not be verified,
2. the above technique is not necessarily water-tight. So the passport holder can still send subliminal information,
3. excluding the use of the subliminal channel by the passport holder enables ***terrorist sponsoring countries to use it***!

We now discuss these problems in more detail.

5.1.1 Problems with verification

The problem is that the solution can not be verified adequately. The Fiat–Shamir protocol (among other applications) is also intended for passport purposes. In such application each country has its own passport center. So the following problem appears: are all countries honest? Indeed will all countries enforce their centers to put a really random source in their hardware Fiat–Shamir passport device, ***even if the centers or some organizations sponsored by the government have all advantages*** not ***to do it***!?

Related to the last questions two possibilities exist. The first one is that the center puts indeed a truly random source in the passports and the second is the opposite. Remark however that as a consequence of the tamperfree aspect, it is impossible to know which is the case. We now discuss these two cases in respectively Section 5.1.2 and Section 5.1.3.

5.1.2 Solution not water-tight

Let us now suppose that indeed a truly random source would be used inside the tamperfree passport. Then the passport holder can still communicate. Hereto he uses his passport. When a certain bit of t corresponds with the first bit of his ciphertext, then he sends this t to the verifier, else he does not send it to the verifier. In the last case he answers himself instead. The passport device is not able to see a difference between an interaction with the real verifier or with the passport holder himself who simulates a verification step. The passport holder continues similarly with the next bits of the ciphertext.

The reader could first remark that the warden will observe that an extra device is put in the middle between the passport device and the verifier, in order to have a filter effect on the outcoming t. If the passport holder is clever, he avoids detection using, *e.g.*, the following technique. He puts a chip on the surface of the passport device. This chip has to filter out the unwanted t, using the described method. He repaints the surface, so that it would have the same appearance. The reader who is now remarking that this will be detected is arguing that documents and/or devices can be made which have an unforgeable appearance. So he is arguing that the actual passports are secure by using unforgeable stamps. But wasn't the motivation for the invention of the Fiat–Shamir scheme not that such unforgeable stamps do not exist?

The reader could also argue that the capacity of the subliminal channel is extremely low. However such channels can also be dangerous [20].

5.1.3 The solution helps terrorist sponsoring countries

Suppose now that the center of βland puts a memory chip in the passport device, instead of a truly random source. The contents of the memory chip corresponds with the $\sqrt{t}$, which will be consecutively used by the passport device. So $\sqrt{t}$ corresponds with consecutive bits stored in the memory chip. The center now claims that they are so friendly to inform the warden what the t are that the device will use. The warden can have the impression that this is indeed a very kindly idea of the center. Indeed the above subliminal channel (see Section 5.1.2) is now no longer possible (without detection), *if* the center does not collaborate with the passport holder. *However* even if it is true that there is *no collaboration*, then the *center itself* is able to *use the subliminal protocol for its own purposes* (or for governmental use).

The center can now indeed abuse the passport(s) by choosing the $\sqrt{t}$ such that t corresponds with the encrypted message C (or a part of it) that the center wants to send! So βland could use the passport system to transmit information to terrorists who live in αland, through their citizens who are not aware of the fraud. The efficiency of this subliminal channel is 50%. If n is about 200 digits, $k = 1$ and the ping-pong protocol is repeated only 30 times, then (using an inefficient source coder) 500 words of information can be sent. *There is no doubt that the identification protocol becomes more dangerous* if the above *"solution" is used*, than without it!

Similar remarks hold for the choice of e.

5.2 A secure solution

By borrowing an idea from [5] we now propose a secure solution. In this solution we do *not* discuss how to avoid a subliminal message inside j or inside the identity. Our solution guarantees that the following identification ping-pong protocol is subliminal-free (for a proof see [6]).

The main idea (originating from [5]) is to use an *active warden* who acts as an *active* eavesdropper. The active warden has to be *trusted* that he will act exactly as described. Indeed if he wants he can modify the transmissions such that a legitimate identity turns out to be rejected by the verifier. We trust the warden that he will not attempt to influence this rejection. In order to guarantee the subliminal-freeness, the active warden must be trusted not to collaborate with the passport holder and not with the verifier (and not with the center who issued the passport) and trusted that he will not introduce himself a subliminal channel. Full detail of these trust aspects can be found in [6].

The active warden could correspond with a special governmental organization of the country which is visited by Alice (αland). So the warden may not correspond with some foreign country, even not the country of citizenship of Alice (βland). If βland insists to be active warden (to avoid that Alice asks politic asylum), then a second warden of αland is necessary (to avoid abuse by βland). Our solution allows such a combination. Evidently the solution we present, has only sense if no extra channels exist, which are not watched by the warden. Let us describe it for the general Fiat–Shamir scheme (similar for the Feige–Fiat–Shamir version).

Step 1 Alice chooses a $\sqrt{t} \pmod{n}$ and sends t to Bob (through the active warden).

Step 2 The active warden receives t from Alice and chooses truly random bits $f_1, f_2, \ldots, f_k$ and chooses a truly random r. The active warden sends then $t * r^2 * \prod_{f_i=1} m_i^{-1} \pmod{n}$ to Bob

Step 3 Bob sends $e_1, e_2, \ldots, e_k$ to Alice (through the active warden).

Step 4 The active warden receives the bits from Bob and sends $e_1 \oplus f_1, e_2 \oplus f_2, \ldots, e_k \oplus f_k$ to Alice (where $\oplus$ is the exlusive or).

Step 5 Alice answers, this means she sends

$$\alpha \equiv \sqrt{t} * \prod_{e_i \oplus f_i = 1} \sqrt{m_i} \pmod{n}$$

to Bob, through the warden.

Step 6 The active warden receives α and *checks it. If* the answer is wrong, *then* the active warden stops the ping-pong protocol and takes Alice into custody. So using this arrest Alice can communicate one bit to Bob, but the price is high! *Else* (when the answer is correct) the active warden sends

$$\beta \equiv \pm \alpha r \prod_{\overline{e_i} \wedge f_i = 1} m_i^{-1} \pmod{n}$$

where $\pm$ is chosen randomly by the warden and where $\overline{e_i}$ corresponds with the logic complement of e_i and where $\wedge$ is the logic "and". The reader can easely verify that β corresponds with the answer that Bob expects.

Step 7 Bob verifies that β is correct.

The necessary proofs can be found in [6]. *We remark that if Bob has indeed chosen his e_i randomly then the zero-knowledge proof convinces two individuals at the same time.*

6 Conclusion

In our paper we came up with several abuses of the (Feige–)Fiat–Shamir protocol. The importance of physical description was analyzed. When the physical description is used appropriately then zero-knowledge techniques are not necessary for secure passports, else some main weaknesses appear, as in the Feige–Fiat–Shamir scheme. These weaknesses can be solved (see [1], [5] and [14]), but these solutions were out of the scope of this paper. The use of subliminal channels inside passports protocols was analyzed and is clearly a major danger. It is infeasible to detect their use. Subliminal channels are also possible and sometimes preventable in other zero-knowledge protocols (see [6]). This can now easely be figured out by the reader who understands all details of the above protocols. This fact allows to think about the fact that "zero-knowledge proofs" and "zero-knowledge about zero-knowledge proofs" are not necessarily forced to be zero-knowledge.

A problem with the (Feige–)Fiat–Shamir protocol and several other identification systems is that eavesdroppers can easely figure out who is travelling and when. They have simply to do the verification on their own.

It becomes clear that the (Feige–)Fiat–Shamir schemes are not the ultimate passport protocols. Other passport protocols having several advantages and one of them also allowing identification over the telephone, are coming up ([5] and [14]). We can expect that in the near future very secure passport systems, which satisfy our current notion about passports and which are doing nothing more or less than identifying and which are more secure and better from all points of view than the ones used today, will appear.

References

[1] S. Bengio, G. Brassard, Y. Desmedt, C. Goutier, and J-J. Quisquater. Aspects and importance of secure implementations of identification systems. June 1987. Submitted to the Journal of Cryptology.

[2] M. Blum and S. Goldwasser. An efficient probabilistic public–key encryption scheme which hides all partial information. In *Advances in Cryptology. Proc. of Crypto'84 (Lecture Notes in Computer Science 196)*, pages 289–299, Springer–Verlag, New York, 1985. Santa Barbara, August 1984.

[3] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo–random bits. *Siam J. Comput.*, 13(4):850–864, November 1984.

[4] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, February 1981.

[5] Y. Desmedt. A subliminal-free authentication system and its use for identification. In preparation.

[6] Y. Desmedt and C. Goutier. Abuses of zero-knowledge proofs, in particular the Fiat-Shamir identification protocol. In preparation.

[7] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat–Shamir passport protocol. February 28, 1987. Submitted version of the paper.

[8] Y. Desmedt and J.–J. Quisquater. Public key systems based on the difficulty of tampering (Is there a difference between DES and RSA?). Presented at CRYPTO'86, Santa Barbara, California, U. S. A., August 11–15, 1986, extended abstract will appear in Advances in Cryptology, Proc. of Crypto'86, Lecture Notes in Computer Science, Springer–Verlag, 1987.

[9] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT–22(6):644–654, November 1976.

[10] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proceedings of the Nineteenth ACM Symp. Theory of Computing, STOC*, pages 210 – 217, May 25–27, 1987.

[11] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. August 3–11, 1986. Presented at the International Congress of Mathematicians, ICM 86, Berkeley, California, U.S.A.

[12] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. August 11–15, 1986. Presented at Crypto'86, Santa Barbara, California.

[13] J. Gleick. A new approach to protecting secrets is discovered. *New York Times*, pp. C1 and C3, February 18, 1987.

[14] J.-J. Quisquater. Signatures, identifications et controles d'accès. December 16, 1986. Lecture given at INRIA (France).

[15] J. Saltzer. On digital signatures. *ACM Operating Syst. Rev.*, 12(2):12 – 14, April 1978.

[16] A. Shamir. Interactive identification. March 23–29, 1986. Presented at the Workshop on Algorithms, Randomness and Complexity, Centre International de Rencontres Mathématiques (CIRM), Luminy (Marseille), France.

[17] G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology. Proc. of Crypto 83*, pages 51–67, Plenum Press N.Y., 1984. Santa Barbara, California, August 1983.

[18] G. J. Simmons. The secure subliminal channel (?). In H. C. Williams, editor, *Advances in Cryptology. Proc. of Crypto 85 (Lecture Notes in Computer Science 218)*, pages 33–41, Springer–Verlag, 1986. Santa Barbara, California, August 18–22, 1985.

[19] G. J. Simmons. A system for verifying user identity and authorization at the point-of sale or access. *Cryptologia*, 8(1):1–21, January 1984.

[20] D. Slater. A note on the relationship between covert channels and application verification. *ACM, SIG Security Audit & Control Review*, 5(1):22, 1987.

Direct Minimum-Knowledge Computations

(Extended Abstract)

Russell Impagliazzo — **U.C. Berkeley**

Moti Yung[1] — **Columbia University**

Abstract

We present a protocol scheme which *directly* simulates any given computation, defined on any computational device, in a *minimum-knowledge* fashion. We also present a scheme for simulation of computation in *dual* (perfect) *minimum-knowledge* fashion. Using the simulation protocol, we can assure that one user transfers to another user exactly the result of a given computation and nothing more.

The simulation is direct and efficient; it extends, simplifies and unifies important recent results which have useful applications in cryptographic protocol design. Our technique can be used to implement several different sorts of transfer of knowledge, including: transfer of computational results, proving possession of information, proving knowledge of knowledge, gradual and adaptive revealing of information, and commitment to input values.

The novelty of the simulation technique is the separation of the data encryption from the encryption of the device's structural (or control) information.

---

[1] Supported in part by NSF grants MCS-8303139 and DCR-8511713 and an IBM graduate fellowship.

1. Introduction

Zero-knowledge interactive proof-systems are a new technique which can be used as a cryptographic tool for designing provably secure protocols. Goldwasser, Micali, and Rackoff originally suggested this technique for controlling the knowledge released in an interactive proof of membership in a language, and for classification of languages [19]. In this approach, knowledge is defined in terms of complexity theory, and a message is said to convey knowledge if it gives a computational advantage to the receiver, for example by giving him the result of an intractable computation. The formal model of interacting machines is described in [19, 15, 17].

A *proof-system* (for a language L) is an interactive protocol by which one user, the *prover*, attempts to convince another user, the *verifier*, that a given input x is in L. We assume that the verifier is a probabilistic machine which is limited to expected polynomial-time computation, while the prover is an unlimited probabilistic machine. (In cryptographic applications the prover has some trapdoor information, or knows the cleartext of a publicly known ciphertext.)

A *correct* proof-system must have the following properties:

- If $x \in L$, the prover will convince the verifier to accept the proof with very high probability.
- If $x \notin L$ no prover, no matter what program it follows, is able to convince the verifier to accept the proof, except with vanishingly small probability.

The above definition can be extended to correctness of a more general protocol which transfers to the verifier the result of a computation. The possible results transferred by the protocol form a probability distribution defined by the computation of the probabilistic machines. Such a protocol is *correct* if the prover can transfer a correct result according to the specified probability distribution (with very high probability), and no adversary can transfer an incorrect result by biasing the distribution (except with a vanishingly small probability).

In order to model the fact that the interacting parties are not memory-less and, for example, may choose their action according to previous events, they are provided with *history* tapes. Each user's history includes the messages exchanged, the random bits used and the private output that he computes during the execution.

A proof-system is *zero knowledge* if the following holds. Given any input $x \in L$, and an initial verifier's history h (representing the context in which the protocol is started), any verifier can generate a probability distribution of possible transcripts of a simulated history of an interaction. The simulation should not be distinguishable by any polynomial time computation from the history of actual interactions (of the prover and this verifier) on the same input x and same initial history h. The definition of *polynomial time indistinguishability* is given in [23, 19]. The zero-knowledge property implies that the interaction gives the verifier the intended result and nothing more, a fact which is crucial in controlling the knowledge transmitted in cryptographic protocols.

The zero-knowledge notion has been useful in designing secure protocols; see for example [19, 13, 11, 2]. In [15] the notion was extended to deal with the knowledge released in a more general protocol which transfers a computational result; an example was given of a protocol for transferring the value of a certain number-theoretic predicate.

For a protocol which transfers a result, we say that the prover gives the computational result to a verifier in a *minimum-knowledge* fashion if the verifier, when he is allowed to get one answer from a *result oracle*, can generate a simulated transcript of the history of an interaction which is indistinguishable from the history of an actual interaction with the prover which gives him this result. This was originally formalized for result transfer protocols in [15]: if the provision of the oracle's answer enables the verifier to simulate the entire interaction, then we can say that the interaction itself did not give any additional knowledge to the verifier.

An important result by Goldreich, Micali and Wigderson [17] shows that, under the assumption that a one-way permutation exists, all NP languages have zero-knowledge proofs. This result is the starting point of our work. As described in [17], this result has important consequences for the design of provably secure protocols, as will be explained in the next section.

The result of [17] is proved by exhibiting a protocol for graph 3-colorability, i.e. a proof-system for the language of 3-colorable graphs. To prove a general NP statement one first has to translate it into an instance of the graph 3-colorability problem (using the publicly known Karp reduction [16]). Recently, several protocols for other NP languages were suggested, using similar techniques which exploit properties of specific NP-complete languages. In [8], under the assumption that quadratic residuosity is intractable, a protocol for satisfiability (SAT) was suggested which directly simulates the circuit that evaluates given instances of SAT. Chaum [10], independently, gave a protocol for direct circuit simulation under the assumption that "claw-free" functions exist. In [17] it is mentioned that the technique of direct computation in [8] does not seem to generalize to an arbitrary one-way function. Furthermore, it seems that the techniques of [10, 8] do not generalize to a direct simulation of a general computational device, since they rely on the input-output relationships of circuit gates.

In this paper we show how it is possible to directly simulate the computation of any given computational device. The scheme presented is correct; if a one-way permutation exists, then the scheme guarantees that the transfer is done in minimum-knowledge fashion. Our construction applies to any polynomial-size device (such as a polynomial-size circuit, or a polynomial-time non-deterministic or probabilistic-non-deterministic Turing machine [20]). The minimum-knowledge property assures that any information about the input which is not revealed by the output remains secure. Using this new technique, one can avoid the Karp reduction to a specific NP-complete problem, and directly prove the result of the computation.

The efficiency of our protocol scheme is directly related to the computational complexity of the given problem. When the computation is given by a computing circuit of size c (or a Turing machine in which the product of time and space is c) and k is the size of an encryption of one bit (which is equal to the system's security parameter), the message size required is ck. In order to make the protocol correct with high probability, we have to pay in communication rounds. If the tolerated error probability of the protocol is $\delta(k)$, then r, the number of rounds in our scheme, must be $\log\delta(k)$. In case one wants an exponentially small probability, then $r=k$; if δ has to vanish faster than any k^{-n} for all n, then $r=\log^{1+\epsilon}k$ is enough. Thus, the total communication complexity is ckr, where k is the overhead for encryption and r is the overhead for confidence.

Our protocol scheme applies to several different notions of "minimum-knowledge computations" suggested recently. It is a uniform method which can be used in the following interactions:

- It enables the prover to simulate a given computation and transfer only its result [19, 15, 17]. The prover can transfer only parts of the computation (parts of the input, output or intermediate results), hiding all other information from the verifier. It seems that exhibiting such partial relations in a minimum-knowledge fashion is harder to do using the reduction of the computation to an NP statement, while in direct computation, partial data decoding is easily achievable, and is minimum-knowledge with respect to a 'result oracle' which gives the verifier these partial data.
- It can be used to convince the verifier that the prover possesses a piece of information. This is similar to what was formalized as proving "possession of information" [22], and proving "knowledge of knowledge" [12].
- It has a significant advantage over NP-reduction in solving the problem introduced here of revealing the output information in an "interactive gradual fashion"; that is, when bits of the results are being revealed gradually one by one by the prover, for example, in exchange for an appropriate payment by the verifier. The protocol is run once and for all, and later only the result bits are opened gradually; no extra interaction is needed. In the full paper we show how to do this and how to perform "adaptive gradual trading of a result", in which a verifier may decide which partial result he wants based on previously opened bits. This adaptive gradual opening of information is suggested and solved here. The difficulty is that, in order to make sure that such an interaction is minimum-knowledge, the simulating verifier should receive only the required information which is actually opened, and in an adaptive fashion.
- It can be used directly to verify a "commitment to a value". This is a model of computation with encrypted data presented by Yao [24]. In this model, a prover commits himself to certain data by announcing an encrypted version of it. Later, he can convince the verifier that a computational result transferred indeed used as inputs the cleartext values to which he had committed himself.
- It can also simulate the most general interactive proof (IP) in a minimum-knowledge fashion. The technique is applied to the probabilistic nondeterministic polynomially bounded Turing machine [20] which models the general interactive proof. The fact that the general interactive proof can be done in minimum-knowledge was first observed by Ben-Or.

The *dual* (or perfect) *zero-knowledge* notion was suggested by Brassard and Crepeau, and by Chaum [7, 10]. The prover is restricted to polynomial time; however he knows a witness to the given NP statement, or an input to the given computation. The verifier may have unlimited computing power (or, in a variation on this model, we assume that he is limited to polynomial-time), and he wants to be convinced that the prover has a witness.

The differences between the dual model and the model described above (which we may call the *primal* model) are:

- The computational power of the parties is interchanged.
- The verifier accepts the proof only under a number-theoretic or cryptographic assumption in the dual model and unconditionally in the primal one.
- The computation simulating the actual interaction in the dual model produces a transcript of interaction which is identical to the original one (and thus indistinguishable); such a protocol is called *perfectly zero-knowledge*. In the primal model the transcript of the interaction is not

identical, and is indistinguishable only under the computational complexity assumption.

- The effect of deviations from the specified computational power of the parties differs in the two models. In the primal model, if the verifier is given enough computational resources before the interaction (say, exponential time), the interaction is still (trivially) minimum-knowledge (using a simulator with the same resources as the verifier). In the dual model, when the prover gets enough time before the interaction, he can cheat and the protocol might not be correct. After the interaction, on the other hand, in the primal model giving enough time to the verifier may enable him to extract more knowledge than intended; in the dual model the verifier cannot extract any extra knowledge when given unlimited time after the interaction.

Assuming there exists a one-way function which is a group homomorphism, we give a direct simulation for the dual minimum-knowledge model. Notice that all known number-theoretic permutations which are assumed to be one-way (based on the problems of RSA, quadratic residuosity, and discrete logarithm) are group homomorphisms. The proof-system protocol for this model is similar to the first simulation protocol; thus we present a unified way to treat the two models. We note that the dual zero-knowledge model protocols in [7, 10] are based on quadratic residues or claw-free pairs of functions, and they require the ability to prove equality of two encrypted bit values in a minimum-knowledge fashion. We do not need this property for our protocol. The only requirement is that the homomorphism is one-way; it is used in an encoding technique we call *locking*. The simulation technique in the dual model applies to all the various notions of "minimum-knowledge computations" presented above which are applicable to this model.

In this abstract we present only the basic protocols for simulation of circuit computations. The precise definitions of the various notions (such as "minimum-knowledge" and "transfer of knowledge") will be given in the full paper.

2. Applications of Zero-Knowledge Computations

We assume that a probabilistic encryption scheme exists. Yao [23] showed that this is implied by the existence of a one-way permutation. Examples of concrete encryption schemes based on assumptions of the intractability of certain number-theoretic problems are given in [18, 3, 4, 1]. In the full paper we formalize exactly what properties are required of the encryption scheme in order to implement our constructions.

When a cryptographic protocol is performed, each user wants to be sure that his partners follow the protocol. For example, in a "mental-poker" game a player wants to know that his opponent follows his instructions as specified by the protocol, and gets legal cards from the deck. The fact that every NP language has a zero-knowledge proof makes on-line validation of such facts possible [17]. Given a probabilistic encryption scheme, the set of valid encryptions of a legal message is an NP language. Thus, users can check interactively in minimum-knowledge fashion that a message sent during an execution of a protocol was indeed generated as specified by the protocol. We call protocols in which each message is checked on-line using zero-knowledge proofs *validated* protocols. This validation can be the bottleneck in the protocol. Therefore, to make validated protocols efficient, the proof-systems used throughout the protocol should be as efficient as possible. Our direct computation technique can speed up the validation proofs.

Here we present a few examples of validations. Assume that a user is supposed to choose a number which is a product of m primes; he constructs a probabilistic circuit which checks primality of m inputs, then multiplies them and gives the product as an output. Afterwards, he can perform a minimum-knowledge simulation of the computation and open only the output.

In another example, suppose that a user sends a value $y=f(x)$ and wants to show that he knows the argument x of the one-way function f, while keeping x secure. He can convince another user by a minimum-knowledge simulation of the circuit which computes $f(\cdot)$. The simulation itself uses a one-way function to encode the circuit, which we may call the underlying encryption function. (In fact, we can use a 'bootstrapping technique' in which the circuit computation that represents the computation of f is performed using f itself as the underlying function). In the next section we explain how these computations are done.

In the above example, the result of the computation is the only information opened by the user. It may be the case that this information has to be opened gradually, and the receiver of the information has to pay for each result bit (for example by revealing bits of his own secret result). This is the "gradual interactive revealing of information" that we introduce in this work (to be explained in the full paper). The notion is similar to the one presented in [9], but we limit the interaction to an initial phase, separate from the actual opening of the bits. We also present an adaptive version where the bits to be revealed are decided on-line by the verifier.

Assume that a relation $Q=(\cdot,\cdot)$ and a public input x are given. A prover wants to demonstrate that he "knows" or possesses [22] a private witness w such that $(x,w)\in Q$. For example, the witness can be a certificate of the membership of x in an NP language. The possession of the witness can be proved using a minimum-knowledge simulation of the computation of the predicate Q. Similarly, the prover can demonstrate that he has a witness w either to the fact that $x\in L$ or to the fact that $x\notin L$, where $L\in$ NP $\cap$ co-NP, without revealing to the verifier which is the case. There is a a predicate $A(x,w)$ for L, and a similar predicate $B(x,w)$ for the complement language $\overline{L}$. The prover has to convince the verifier that he has a witness which satisfies the predicate $A(x,w)\vee B(x,w)$; he does this by executing a minimum-knowledge simulation of the computation of this predicate.

Suppose a prover wants to demonstrate that he computes $C(x)$ with a value x that he knows, and to which he has committed himself [24]. The commitment is done at the beginning of the protocol, when the prover sends y, where $y=f(x)$ is the image of x under a one-way function. Then, using the same input x, he evaluates both $f(\cdot)$ and $C(\cdot)$. The prover opens the outputs, which have to be y and the result $C(x)$. The verifier is convinced that the computation must have used the input to which the prover committed himself earlier.

3. Direct Minimum-Knowledge Computations

In this section, we show how any one-way permutation may be used in *direct* minimum-knowledge interactive simulation of a computation. Let P be the prover and V the verifier.

The essential idea of the protocol is that the prover constructs and sends to the verifier a copy of a simulation of the computing device (i.e. a copy of a circuit or of a computation history of a Turing machine). This copy includes encoding of the possible input, intermediate results, and output data. In addition it includes encoding of structural information about the computing device. In the case of a circuit, the structural information consists of connections (pointers) from the output of one gate to the input of another gate in the circuit (based on the gates' truth tables); a pointer connects two entries which encode the same bit value. For a Turing machine, the structural information consists of connections between two consecutive instantaneous descriptions in the computation history; these connections are based on the machine's finite control (transition table). Any computation is a combination of some local data manipulation and transitions, based on the device's finite control, which direct the computation to its next step. Our technique separates the encoding of data from the encoding of the possible continuations given by the control element. We describe here only the circuit model computations.

First we outline the protocol. Upon receiving an encoding of the circuit, the verifier flips a coin and chooses one of two options. With probability 1/2 he decides to *verify*, that is to request that the prover open all the encryptions in the copy of the circuit. In this case he can check that the construction is a legal computing circuit with the right data in each gate, and that the structural information encodes correct connections between gates. With probability 1/2 the verifier chooses to *compute*, in which case the prover opens only the result of the computation. To prove that the output presented is in fact the computed result of the circuit, the prover opens only the cleartexts of the pointers connecting entries which are involved in the computation, while all other information is left encrypted. The connections show how to navigate through the circuit from the input data to the output gates. Then the circuit's output is opened as well. The unopened information appears random (and is polynomial-time indistinguishable from a set of encryptions of any fixed arbitrary data).

The process is repeated r times; each time the prover is ready either to verify or to compute. If all verifications are successful (that is, each circuit encoding does simulate the computing circuit), and all the computations produce a connection to the same opened output, then the verifier accepts the result of the computation.

In a computation, only pointers between gates leading to the output are opened. We remark that if there are random inputs in the circuit, then from the two possible elements in an input entry V chooses at random which element to use, and P uses these elements in the computation. (V is actually flipping a coin into P's well [5].) In the direct simulation, the result of the computation need not be restricted to an output of the given circuit. It can include relations between parts of the inputs, intermediate results, and outputs.

Below we briefly describe the circuit design. (We will give a detailed description in the next version.)

- A circuit encoding consists of gates corresponding to the actual gates of the given circuit.
- Each gate has input entries, a truth table and output pointers.
- Each input entry represents an input bit to the gate and consists of an unordered pair of ciphertexts encrypting 0,1 or 1,0 at random. In an actual computation one of the two ciphertexts is chosen.
- The truth table represents the computation done by the gate. It consists of rows; each one maps a combination of entries' ciphertext values to an output pointer. (For example, a binary

gate has four rows.)

- The rows of each truth table are permuted at random.
- The pointer in each row is a reference to the input entry of the next gate in the circuit. The pointer value is either *first* (an encryption of 0) or *second* (an encryption of 1).
- If the pointer value is *first* (0) it means that the output value of this row is the same as the value of the first ciphertext in the input entry of the next gate, while a pointer of value *second* (1) means that the output is the same as the second value in the input entry of the next gate.
- Since the entries and rows in the table are randomly permuted in each gate, the pointer connection giving the structural information about the circuit appears random when it is deciphered while keeping the other information encrypted.
- If the gate does not connect to another gate but actually gives a circuit output bit, then the pointer's value (i.e. the bit it encrypts) is the actual output value.

Next we briefly describe the protocol.

PROTOCOL 1:

{The prover P (who uses his probabilistic encryption procedure E) and the verifier V simulate the computation of a circuit C in a minimum-knowledge fashion.}

repeat r times <u>{loop}</u>:

1. P probabilistically encrypts C as described above. Let E(C) denote the encryption. $P \rightarrow V$: "E(C)".

2. V chooses a bit $b \in \{0,1\}$. $V \rightarrow P$: "b".

3. If b=0 then

{*verify*}: P opens all the encryptions of all gates in E(C) and sends the cleartext circuit to V.

else

{*compute*}: P opens only the pointers of the specific computation and sends the cleartexts of these pointers to V (including the outputs).

4. If b=0 V verifies that C is properly encrypted;

if b=1 he verifies that the opened pointers lead from the (unopened) input entries to the output pointers.

<u>{end loop}</u>

If all computations give the same value and all verifications are successful then V accepts. In any other case he rejects.

{END PROTOCOL 1}

Theorem 1: Assume that a one-way permutation exists. There is a direct minimum-knowledge simulation of any circuit of polynomial size c. It takes r rounds and uses kc-bit messages, where k is the security parameter and c is the size of the simulated circuit. The error probability is $1-(1/2^r)$.

The protocol is a correct result-transfer protocol because:

- The prover is able to transfer the result and compute the encrypted circuit transferring the result.
- On the other hand, no prover can cheat the verifier since in order to do this without being caught he has to guess the r random coins of step 2 and this event has vanishingly small probability ($1/2^r$).

Notice that with very high probability ($1 - 1/2^r$) any machine P' that successfully performs the prover's role is ready, in at least one of the iterations, to verify that indeed he constructed the circuit simulation properly, but is asked instead to compute. This means that in at least one of the rounds the prover indeed had a valid circuit and indeed computed using a witness. Since all computations and verifications were valid, the verifier accepts the correctness of the computation. This is true also when the computational result is actually not interesting, but the circuit is used only for a demonstration of some computational power as was formalized in [12, 22] (e.g. "knowledge of a witness" as in one of the examples of section 2).

The system is proven to be minimum-knowledge by showing that for any given verifier V' there is a simulating polynomial-time machine. The machine gets the result of the computation from a 'result oracle' (as was described in [15]) and produces (in expected polynomial time) an output which is a simulated history of the interaction; this output is polynomial-time indistinguishable from a history of V' recorded in an actual interaction between V' and the prover.

The crux of the proof is that the unopened part of an actual interaction is indistinguishable from an encryption of fixed random data in the simulated transcript. No polynomial time procedure can get any partial information about this unopened ciphertext of the transcript in order to tell whether it is an actual or a simulated one. If there is a polynomial time procedure which does, then the set of transcripts is a message space for which it is possible to distinguish between encrypted messages in polynomial time. These messages are probabilistically encrypted based on a one-way function. Such a procedure could be converted into a procedure which efficiently inverts the one-way function, but this is assumed to be impossible [18, 23, 21].

4. Direct Dual (Perfect) Minimum-Knowledge Computations

The direct minimum-knowledge computation technique presented is also applicable to the dual model of minimum-knowledge protocols. In this model the proof is presented by a polynomial-time prover to a powerful verifier, and the proof is accepted only under a cryptographic assumption [7, 10]. The proof has to convince the verifier that the prover possess some computational knowledge. Chaum [10] uses this model for identification protocols in the context of his credential mechanism.

Assume there is a one-way function f which is a group homomorphism. (All the number-theoretic one-way functions discussed in the literature are group homomorphisms: RSA, modular exponentiation, modular squaring). Then the following holds:

- Given a random ciphertext value $f(x)$, no polynomial time procedure can find x. This follows from the assumption that the function is one-way.
- Given a random ciphertext value $f(x)$, and a random value q in the range of f. It is impossible

(even with unlimited computing power) to decide whether q was generated as $q=f(x)*f(s)$ for some randomly chosen s, or $q=f(t)$ for some randomly chosen t.

- A random polynomial time procedure which, on input $f(x)$, is able to compute q, s, and t related as above, is able to compute x as well. This would contradict the assumption that f is one-way.

These facts are the basis of the following protocol scheme. Previously, the dual minimum-knowledge simulated computation protocols [10, 7, 6] needed the ability to provide a *zero-knowledge proof of equality* of two cleartexts, given the ciphertexts. Here we show that, using the new technique, this requirement is not needed.

PROTOCOL 2:

{The polynomial time prover P and the verifier V simulate the computation of a circuit C.}

1.V $\rightarrow$ P: " $z=f(x)$ ".

{It is possible to have a model of computation in which z is given and x is unknown to both parties. On the other hand, it might be the case that the verifier has unlimited power and knows x, in which case the above transmission can be followed by a minimum-knowledge proof in which V validates that the value z is in the range of f and x is kept secret. This is done using the zero-knowledge proof of a preimage of a group homomorphism of [14] (generalizing the proof of quadratic residuosity of [19]).}

P encrypts C as described in section 3 above, using as encryption function the following:
0 is encoded as $f(t)$, for a randomly chosen t,
1 is encrypted as $z*f(s)$, for a randomly chosen s.

{Notice that in the above encryption methods, the encodings of 1 and 0 are drawn from the same probability distributions. The powerful machine V cannot distinguish between an encryption of 0 and an encryption of 1. The only difference between them is the way they are produced by the prover. We call this encryption method *locking*, since it is the computation by P which commits the ciphertext to a value, and he is the only one who can unlock and reveal the original value.}

{Exhibiting a cleartext value for a ciphertext y is done by opening (*unlocking*), which means giving either a preimage by f of y (to unlock y as an encryption of 0) or a preimage of $y*z^{-1}$ (to unlock y as an encryption of 1).}

2. Using the locking of bits as encryption mechanism, the users follow the loop and the acceptance procedure of protocol 1.

{END PROTOCOL 2}

Theorem 2: Assume that a one-way group homomorphism exists. There is a direct 'dual zero-knowledge' simulation of any circuit of size c. It takes r rounds and uses kc-bit messages, where k is the security parameter. Its error probability is $1-(1/2^r)$.

The proof relies on the properties of the locking method described above. The fact that locked 0 and locked 1 are identical (absolutely indistinguishable) makes the set of transcripts output by the simulating machine *identical* to the set of histories of a real interaction between a verifier and the prover. Such a protocol is called "perfectly minimum-knowledge".

The verifier accepts the proof under the assumption that the prover cannot invert $f(x)$, and when unlocking a value, say 1, he can open it only as 1. Notice that as in the proof of theorem 1, with very high probability, in one of the iterations a prover P' is ready to verify the circuit construction, but is asked, instead, to compute; thus V accepts with very high probability. A prover cannot cheat since he is likely to be caught with very high probability. This shows that the protocol is correct.

5. Conclusions

Minimum-knowledge interaction seems to be an important tool for the implementation of secure correct protocols. We presented direct minimum-knowledge interactive computation systems. Our implementation is easily derived from the specification of the computational problem; it is simple and efficient. It is appropriate for all of the models of minimum-knowledge interaction that have recently been proposed, as well as the new ones proposed here.

The technique of direct computation deals in a uniform way with any computational problem, and applies uniformly to the two models of zero-knowledge interactions and to the various notions of interactive knowledge-transfer.

Acknowledgments

We wish to thank Manuel Blum, Gilles Brassard, David Chaum, Oded Goldreich, Stuart Haber, Steven Rudich, and Mike Sipser for their helpful discussions and comments.

References

1. Alexi, W., Chor, B., Goldreich O. and Schnorr C.P. RSA/Rabin Bits are 1/2 + (1/poly(k)) Secure. Proc. 25th FOCS, IEEE, 1984, pp. 449-457.

2. Benaloh, J.C. and Yung M. Distributing the Power of a Government to Enhance the Privacy of Voters. Proc. 5th PODC, ACM, 1986, pp. 52-62.

3. Blum, M. and S. Goldwasser. An Efficient Probabilistic Public-Key Scheme Which Hides All Partial Information. Proceedings of Crypto84, 1985, pp. 289-301.

4. Blum, L., Blum M. and Shub M. Comparison of Two Pseudo-Random Number Generators. Proceedings of Crypto82, August, 1982, pp. 61-78.

5. Blum, M. Coin Flipping by Phone. COMPCON, IEEE, 1982, pp. 133-137.

6. Boyar, J.F., M.W. Krentel, and S.A. Kurtz. A Discrete Logarithm Implementation of Zero-Knowledge Blobs. 87-002, University of Chicago, March, 1987.

7. Brassard, G. and C. Crepeau. Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond. 27th FOCS, IEEE, October, 1986, pp. 188-195.

8. Brassard, G., and Crepeau C. Zero-Knowledge Simulation of Boolean Circuits. Proceedings of Crypto 86, 1986.

9. Brickell, E.F., D. Chaum, I. Damgard, and J. van de Graaf. Gradual and Verifiable Release of a Secret. These proceedings.

10. Chaum, D. Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How. Proceedings of Crypto86, 1986.

11. Cohen, J.C. (Benaloh) and Fischer M.J. A Robust and Verifiable Cryptographically Secure Election Scheme. Proc. 26th FOCS, IEEE, 1985, pp. 372-383.

12. Feige, U., A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity. 19th STOC, 1986, pp. 210-217.

13. Fischer, M., S. Micali, C. Rackoff, and D. Wittenberg. An Oblivious Transfer Protocol Equivalent to Factoring. Manuscript, 1986.

14. Galil, Z. , Haber S. and Yung M. Symmetric Public-Key Encryption. Crypto85 proceedings, 1985, pp. 128-137.

15. Galil, Z., Haber S. and Yung M. A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems. Proc. 26th FOCS, IEEE, 1985, pp. 360-371.

16. Garey, M.R., and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

17. Goldreich, O., S. Micali and A. Wigderson. Proofs that Yield Nothing But their Validity and a Methodology of Cryptogrphic Protocol Design. 27th FOCS, IEEE, October, 1986, pp. 174-187.

18. Goldwasser, S. and Micali S. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. Proceedings of the 14th Annual ACM Symp. on Theory of Computing, ACM-SIGACT, May, 1982, pp. 365-377.

19. Goldwasser, S., S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. 17 STOC, ACM-SIGACT, May, 1985, pp. 291-304.

20. Goldwasser, S. and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. Proceedings of the 18 Annual ACM Symp. on Theory of Computing, ACM-SIGACT, May, 1986, pp. 59-68.

21. Micali, S., C. Rackoff and B. Sloan. The Notion of Security for Probabilistic Cryptosystems. Proceedings of Crypto86, 1986.

22. Tompa, M. and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information. 28th FOCS, 1986.

23. Yao, A. Theory and Applications of Trapdoor Functions. 23rd FOCS, IEEE, November, 1982, pp. 80-91.

24. Yao, A. How to Generate and Exchange Secrets. 27th FOCS, IEEE, October, 1986, pp. 162-167.

NON-INTERACTIVE ZERO-KNOWLEDGE PROOF SYSTEMS.

Alfredo De Santis †
Silvio Micali *
Giuseppe Persiano †

Abstract.

The intriguing notion of a Zero-Knowledge Proof System has been introduced by Goldwasser, Micali and Rackoff [GMR] and its wide applicability has been demonstrated by Goldreich, Micali and Wigderson [GMW1]-[GMW2].

Based on complexity theoretic assumptions, Zero-Knowledge Proof Systems exist, provided that

(i) The prover and the verifier are allowed to talk back and forth.
(ii) The verifier is allowed to flip coins whose result the prover cannot see.

Blum, Feldman and Micali [BFM] have recently shown that, based on specific complexity theoretic assumption (the computational difficulty of distinguishing products of two primes from those product of three primes), both the requirements (i) and (ii) above are not necessary to the existence of Zero-Knowledge Proof Systems. Instead of (i), it is enough for the prover only to talk and for the verifier only to listen. Instead of (ii), it is enough that both the prover and verifier share a randomly selected string.

We strengthen their result by showing that Non-Interactive Zero-Knowledge Proof Systems exist based on the weaker and well-known assumption that quadratic residuosity is hard.

† Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84100 Salerno, Italy.
* MIT, Laboratory for Computer Science, Cambridge, Mass. 02139. Supported by NSF grant DCR-8413577.

1. Introduction.

In many scenarios, like cryptographic ones, "knowledge" is the most valuable resource and thus one may not want to give away more knowledge than "absolutely needed".

This has been formalized by Goldwasser, Micali and Rackoff [GMR] who introduced the somewhat paradoxical notion of a *Zero-Knowledge Proof System (ZKPS)*. Since then other definitions of Zero-Knowledge has been given by [GHY] and [BC].

This is a special way of proving theorems. It allows A, who holds the proof of a given theorem $\mathcal{T}$, to convince a poly-bounded B that $\mathcal{T}$ is true without revealing any additional information. In other words, in a ZKPS A allows B to verify that a theorem $\mathcal{T}$ is true; however, he does not allow B, after verifying the proof, to compute more than he could have computed after being only told (without any proof) that $\mathcal{T}$ was indeed true.

Under the assumption that one-way functions exist, Goldreich, Micali and Wigderson [GMW1] show that membership in any language in NP can be proved in Zero-Knowledge. ZKPS is indeed a powerful notion and have had a strong impact in the field of cryptographic protocols; see [FFS], [FMRW] and most notably, the recent completeness theorem for protocols with honest majority of [GMW2].

1.1 The communication model needed for Zero-Knowledge.

ZKPS's have been defined and have been shown to exist for a particular communication model between "prover" and "verifier": the interactive Turing machine model of [GMR]. The salient features of this model of communication are the following two:

(i) *Interaction.* The prover and the verifier are allowed to talk back and forth.
(ii) *Privacy.* The verifier is allowed to flip coins whose result the prover cannot see.

Though interaction and privacy of computation are requirements that can be met, in practice they may not be readily available. For example this is the case if A will be away for 10 years and the mail is the only way to communicate with B. During this trip, each time A will succeed in proving a new theorem, he may wish to prove it in Zero-Knowledge to B by simply sending B a letter to which B need not to reply. Thus

Is A's wish possible?

This question has been addressed by Blum, Micali and Feldman [BMF]. Rephrasing it in a more theoretical way:

What are the essential communication features
that make Zero-Knowledge proofs possible?

Assuming that quadratic residuosity is hard, an assumption widely used in Cryptography [GM], we show that ZKPS's exist provided that

the prover and the verifier share
(i.e. have as input a common)
random enough string.

The communication model consisting of sharing a random string was proposed in [BFM]. They showed that in this model ZKPS's exist if it is difficult to distinguish numbers product of two primes from those product of three primes. Our result improves on theirs as the Quadratic Residuosity Assumption is weaker than the assumption used in [BFM]. In fact, if it is easy to distinguish a quadratic residue from a quadratic non residue then it is also easy to distinguish a number product of two primes from one product of three primes, while the converse is not known to be true. Our result is also an improvement in that our algorithm is simpler.

Notice that sharing a random string is a weaker communication model than the one proposed by [GMR]. In fact, in the [GMR] model, prover and verifier may agree on a random enough string by using a coin-flipping protocol [B]. Actually, the new communication model, in a sense explained in section 3.1, is the *minimal* one supporting ZKPS's.

The new model dispenses with the need of privacy of coin tosses as the verifier does not need to use more randomness than the one contained in the common string. Moreover, it dispenses with the need of interaction as well. Once the prover and the verifier share a random string σ, to prove that $x \in L \in NP$, it is enough for the prover to compute a proof (string), $Proof_{x,\sigma}$, in a special way on input x and σ, and send it to the verifier. The verifier, computing on input σ, x and $Proof_{x,\sigma}$ will correctly check that $x \in L$ while receiving zero additional knowledge.

Let us rephrase this in terms of our previous example, where the prover A was leaving for a long trip. Here, if A and B, before A leaves for his long trip, have witnessed some random events, (e.g. a lottery, the sun stains, ...) or have consulted the RAND tables, then A can prove a theorem in Zero-Knowledge just by writing a letter to which B need not to reply.

Let us now proceed more formally.

2. Preliminaries.

2.1 Notations and conventions.

Let us quickly recall the standard notation of [GoMiRi].

We emphasize the number of inputs received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$", if it receives two inputs we write "$A(\cdot,\cdot)$" and so on.

If $A(\cdot)$ is a probabilistic algorithm, then for any input x, the notation $A(x)$ refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ. If S is a probability space, then $Pr_S(e)$ denotes the probability that S associates with the element e.

If $f(\cdot)$ and $g(\cdot,\dots,\cdot)$ are probabilistic algorithms then $f(g(\cdot,\dots,\cdot))$ is the probabilistic algorithm obtained by composing f and g (i.e. running f on g's output). For any inputs $x, y, \dots$ the associated probability space is denoted by $f(g(x,y,\dots))$.

If S is any probability space, then $x \leftarrow S$ denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation $x \leftarrow F$ denotes the algorithm which assigns to x an element selected according to the probability space whose sample space is F and uniform probability distribution on the sample points.

The notation $Pr(x \leftarrow S; y \leftarrow T; \dots : p(x,y,..))$ denotes the probability that the predicate $p(x,y,\dots)$ will be true after the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, \dots$

The notation $\{x \leftarrow S; y \leftarrow T; \dots : (x,y,\dots)\}$ denotes the probability space over $\{(x,y,\dots)\}$ generated by the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, \dots$.

Let us recall the basic definitions of [GMR]. We address the reader to the original paper for motivation, interpretation and justification of these definitions.

Let $U = \{U(x)\}$ be a family of random variables taking values in $\{0,1\}^*$, with the parameter x ranging in $\{0,1\}^*$. $U = \{U(x)\}$ is called poly-bounded family of random variables, if, for some constant $e \in \mathcal{N}$, all random variables $U(x) \in U$ assign positive probability only to strings whose length is exactly $|x|^e$.

Let $C = \{C_x\}$ be a poly-size family of Boolean circuits, that is, for some constants $c, d > 0$, all C_x have one Boolean output and at most $|x|^c$ gates and $|x|^d$ inputs. In the following, when we say that a random string, chosen according to $U(x)$, where $\{U(x)\}$ is a poly-bounded family of random variables, is given as input to C_x, we assume that the length of the strings that are assigned positive probability by $U(x)$ equals the number of boolean inputs of C_x.

Definition (Indistinguishability). *Let $L \subset \{0,1\}^*$ be a language. Two poly-bounded families of random variables $U = \{U(x)\}$ and $V = \{V(x)\}$ are indistinguishable on L if for all poly-size families of circuits $C = \{C_x\}$,*

$$\Big|Pr(a \leftarrow U(x) : C_x(a) = 1) - Pr(a \leftarrow V(x) : C_x(a) = 1)\Big| < |x|^{-O(1)}$$

with $x \in L$.

Definition (Approximability). *Let $L \subset \{0,1\}^*$ be a language. A family of random variables $U = \{U(x)\}$ is approximable on L if there exists a Probabilistic Turing Machine M, running in expected polynomial time, such that the families $\{U(x)\}$ and $\{M(x)\}$ are indistinguishable on L.*

2.2 Number theory.

Let $\mathcal{N}$ denote the natural numbers, $x \in \mathcal{N}$, $Z_x^* = \{y \mid 1 \leq y < x, \quad gcd(x,y) = 1\}$ and $Z_x^{+1} = \{y \in Z_x^* | (y \mid x) = +1\}$, where $(y \mid x)$ is the Jacobi symbol. We say that $y \in Z_x^*$ is a quadratic residue modulo x iff there is $w \in Z_x^*$ such that $w^2 \equiv y \bmod x$. If this is not the case we call w a quadratic non residue modulo x.

Define the quadratic residuosity predicate to be

$$\mathcal{Q}_x(y) = \begin{cases} 0, & \text{if } y \text{ is a quadratic residue modulo } x; \\ 1, & \text{otherwise;} \end{cases}$$

and the languages QR and QNR as

$$QR = \{(y,x) | \mathcal{Q}_x(y) = 0\}$$

$$QNR = \{(y,x) | y \in Z_x^{+1} \text{ and } \mathcal{Q}_x(y) = 1\}.$$

Fact 1: For each $y_1, y_2 \in Z_x^{+1}$ one has

$$\mathcal{Q}_x(y_1 y_2) = \mathcal{Q}_x(y_1) \oplus \mathcal{Q}_x(y_2).$$

Let $Z_s(n)$ denote the set of n-bit integers product of $s \geq 1$ distinct primes. In the following we will use the:

Quadratic Residuosity Assumption (QRA). *For each poly-size family of circuits* $\{C_n \mid n \in \mathcal{N}\}$,

$$Pr\big(x \leftarrow Z_2(n); y \leftarrow Z_x^{+1}: C_n(x,y) = \mathcal{Q}_x(y)\big) < 1/2 + 1/n^{-O(1)}.$$

The QRA is widely used in Cryptography, see for example [GM].

The current fastest algorithm to compute $\mathcal{Q}_x(y)$ is to first factor x and then compute $\mathcal{Q}_x(y)$, while it is well known that, given the factorization of x, $\mathcal{Q}_x(y)$ can be computed in $O(|x|^3)$ steps. Therefore it is usual to choose x product of two large primes of the same length since these integers constitute the hardest input for a factoring algorithm.

3. The main results.

To prove the existence of Non-Interactive Zero-Knowledge Proof Systems for all NP languages, it is enough to prove it for the NP-complete language $3SAT$ [GJ]. For $k > 0$, we define the language $3SAT_k = \{x \in 3SAT \mid |x| \leq k\}$.

We present our result first in a weaker form, for simplicity sake.

3.1 A first solution.

Here we show that if a n^3-bit long string is randomly selected and given to both parties, then the prover can show that any *single* $x \in 3SAT_n$ is indeed satisfiable. Notice that this is not as general as what we promised in the introduction. Only in section 3.2 we will show that, for each polynomial $Q(\cdot)$, using the *same* randomly chosen n^3-bit long σ, any $Q(n)$ formulae $x_1, \ldots, x_{Q(n)} \in 3SAT_n$ can be shown to be satisfiable in Zero-Knowledge.

In the following we formally define the Single-Theorem Non-Interactive ZKPS.

Definition. *A Single-Theorem Non-Interactive ZKPS is a pair (A,B) where A is a Probabilistic Turing Machine and* $B(\cdot,\cdot,\cdot)$ *is a deterministic algorithm running in time polynomial in the length of its first input, such that:*

1) **Completeness.** *(The probability of succeeding in proving a true theorem is overwhelming.)*

   $\exists c > 0$ *such that* $\forall x \in 3SAT_n$

$$Pr(\sigma \leftarrow \{0,1\}^{n^c}; y \leftarrow A(\sigma, x): B(x,y,\sigma) = 1) > 1 - n^{-O(1)}.$$

2) **Soundness.** *(The probability of succeeding in proving a false theorem is negligible.)* $\exists c > 0$ *such that* $\forall x \notin 3SAT_n$ *and for each Probabilistic Turing Machine* A'

$$Pr(\sigma \leftarrow \{0,1\}^{n^c}; y \leftarrow A'(\sigma, x) \colon B(x, y, \sigma) = 1) < n^{-O(1)}.$$

3) **Zero-Knowledge.** *(The proof gives no information but the validity of the theorem.)* $\exists c > 0$ *such that the family of random variables* $V = \{V(x)\}$ *where*

$$V(x) = \{\sigma \leftarrow \{0,1\}^{|x|^c}; y \leftarrow A(\sigma, x) \colon (\sigma, y)\}$$

is approximable over $3SAT$.

Comment. As promised in the introduction we give evidence that our model is the minimal supporting Zero-Knowledge.

The simplest communication model is certainly NP. In it the verifier is deterministic and thus needs not to worry about the privacy of his (non existent) coin tosses. Moreover the interaction is absolutely elementary: on input $x \in L \in NP$, the prover has to talk only once and the verifier has only to listen. What the prover says is w_x, a witness that $x \in L$. Unfortunately, such simplicity does not support Zero-Knowledge. It is not hard to prove that if a language L possesses an NP Proof System that is Zero-Knowledge, then $L \in P$.

Consider a probabilistic version of NP in which the verifier checks that w_x is a proper witness in probabilistic poly-time. Then it is easy to prove that if L possessed such a proof system that is Zero-Knowledge, L would belong to BPP.

So if the prover and the verifier are completely independent, Zero-Knowledge proofs are not possible. In order to have Zero-Knowledge proofs, the prover and the verifier must have something random in common and the simplest way to do this is sharing a random string.

In the following we exhibit a Single-Theorem Non-Interactive ZKPS. We first present informally our protocol and then we prove the following

Theorem 1. *Under the QRA, there exists a Single-Theorem Non-Interactive ZKPS (A,B).*

An informal view of our protocol.

Let $\sigma = \sigma_0 \ldots \sigma_{n^3+7n^2-1}$ be a random string shared by A and B.

Let $C = \{c_1, c_2, \ldots, c_m\}$, $C \in 3SAT_n$, a collection of clauses on the set $U = \{u_1, u_2, \ldots, u_k\}$ of boolean variables. A wants to prove to B that there is a truth assignment that satisfies all the clauses in C, without yielding any additional information. Let $t: U \to \{T, F\}$ be a truth assignment satisfying C.

A's proof consists of two parts: first he exhibits a pair (x, y) such that $x \in Z_2(n)$ and $(y, x) \in QNR$, then he proves $C \in 3SAT$.

A chooses a random $x \in Z_2(n)$, and divides the first $7n^2$ bits of σ in $7n$ n-bit strings, thus obtaining $7n$ integers. A discards those integers not in Z_x^{+1} and partitions the remaining integers in two equivalence classes (one formed of residues and one formed of non residues). To show that two integers, a and b, are in the same class, that is $\mathcal{Q}_x(a) = \mathcal{Q}_x(b)$, A exhibits a random square root modulo x of ab.

A then, chooses two random elements, y_0, y_1, one from each class, if any, and computes $y = y_0 y_1 \bmod x$. Note that $\mathcal{Q}_x(y_0) \neq \mathcal{Q}_x(y_1)$ and so, by Fact 1, y is a non quadratic residue modulo x.

B is now convinced that $x \in Z_2(n)$ and $(y, x) \in QNR$.

A associates an element $w_j \in Z_x^{+1}$ to each literal u_j in such a way that w_j is a non quadratic residue modulo x iff u_j is true under t. To make this association, for each variable u_j, A chooses a random $r_j \in Z_x^*$. Then if u_j is true under t, A associates $r_j^2 \bmod x$ to the literal u_j and $yr_j^2 \bmod x$ to $\overline{u}_j$. If u_j is false under t, A associates $yr_j^2 \bmod x$ to the literal u_j and $y^2 r_j^2 \bmod x$ to $\overline{u}_j$. At this point B can check that, for each variable, the value associated to u_j is a quadratic residue iff the value associated to $\overline{u}_j$ is a non quadratic residue and so he is sure that exactly one of the two literals u_j and $\overline{u}_j$ is true under t.

In this way a triple of values is naturally associated to each clause and A has to prove that each triple is not formed by 3 quadratic residues, that is each clause is satisfied by t.

A divides the last n^3 bits of σ in n sets formed of n strings of n-bit, thus obtaining n sets of n integers. Each set is associated to a clause.

For each clause, A discards from the associated set the integers not in Z_x^{+1} and groups the remaining integers in triple. Since each element of a triple can be a quadratic residue or not, the triples obtained can be partitioned in 8 equivalence classes, that must be roughly of the same size since $x \in Z_2(n)$.

A computes the partition and shows that it is correct in the following way. To show that the tern (a, b, c) is in the same class as (d, e, f), A exhibits random square roots modulo x of ad, be, cf. B can check the correctness of the step, but does not know which kind of triples are in any equivalence class. A can show the class consisting of the terns formed by three quadratic residues by simply disclosing their square roots modulo x.

Finally, let (h, l, m) be the tern associated to the clause c_i. A shows that (h, l, m) belongs to one of the remaining classes, by disclosing three square roots as done above. B can easily check the correctness of the last step. Moreover B knows nothing but that the triple (h, l, m) is not formed by three quadratic residues.

Proof (of theorem 1).

We start by formally describing the protocol (A,B).

A's protocol.

When we say "A writes τ", we mean that A appends τ followed by a special symbol, such as #, to the string $Proof$ that will be sent to B.

Let $\sigma = \sigma_0 \ldots \sigma_{n^3+7n^2-1}$.

Let $C = \{c_1, c_2, \ldots, c_m\}$, $C \in 3SAT_n$, a collection of clauses on the set $U = \{u_1, u_2, \ldots, u_k\}$ of boolean variables. A wants to prove B that there is a truth assignment that satisfies all the clauses in C, without yielding any additional information. Let $t: U \rightarrow \{T, F\}$ be a truth assignment satisfying C.

1) A sets $Proof$ = empty string and writes C.
2) A chooses a random $x \in Z_2(n)$ and writes it.
   A sets E_0, E_1= empty set.
3) For $i = 1, \ldots, 7n$. Let s_i be the integer whose binary representation is $\sigma_{(i-1)n} \cdots \sigma_{in-1}$. If $s_i \notin Z_x^{+1}$ then A discards s_i. If in E_j, $j \in \{0, 1\}$, there is an element s such that $(ss_i, x) \in QR$ then A writes (j, s, γ), where γ is a random square root modulo x of ss_i, and puts s_i in E_j. Otherwise A puts s_i in one of the empty sets, E_j, and writes $(j, 0, s_i)$.
4) If E_0 or E_1 (or both) is empty then A halts.

Otherwise, A randomly chooses $y_0 \in E_0$ and $y_1 \in E_1$, computes $y = y_0 y_1 \bmod x$ and writes (y, y_0, y_1).

5) A picks at random $r_j \in Z_x^*$, $j = 1, \ldots, k$.
If $t(u_j) = F$ A sets $w_j = r_j^2 \bmod x$, otherwise A sets $w_j = yr_j^2 \bmod x$. A writes $(w_1, w_2, \ldots, w_k)$.

6) For each clause c_i, $i = 1, \ldots m$, A performs steps 6.1-6.6.

6.1) Let the clause c_i consist of literals $z_{i_1}, z_{i_2}, z_{i_3}$. If the literal z_{i_j}, $j = 1,2,3$, is the variable u_l then A sets $\eta_{i_j} = w_l$. If the literal z_{i_j} is the complement of the variable u_l then A sets $\eta_{i_j} = yw_l \bmod x$.

6.2) A sets ST = empty stack and repeats n times step 6.3.

6.3) For $j = 0, \ldots, n-1$. Let z be the element whose binary representation is $\sigma_{7n^2+(i-1)n^2+(j-1)n} \cdots \sigma_{7n^2+(i-1)n^2+jn-1}$.
If $z \notin Z_x^{+1}$ then A discards z, otherwise A puts z in the stack ST.

6.4) A sets E_s = empty set, $s = 1, \ldots, 7$.
A repeats step 6.5 until the number of elements in the stack ST is less than 3.

6.5) A picks three elements $\alpha_1, \alpha_2, \alpha_3$ from the stack ST. If $\alpha_1, \alpha_2, \alpha_3$ are quadratic residue modulo x, then A sets γ_j = a random square root of α_j modulo x, $j = 1,2,3$, and writes $(0, \alpha_1, \alpha_2, \alpha_3, \gamma_1, \gamma_2, \gamma_3)$. Otherwise, if there is a triple $(\beta_1, \beta_2, \beta_3)$ in the set E_s, $1 \leq s \leq 7$, such that $\alpha_1\beta_1$, $\alpha_2\beta_2$, $\alpha_3\beta_3$ are quadratic residue modulo x then A puts $(\alpha_1, \alpha_2, \alpha_3)$ in the set E_s, sets γ_j = a random square root of $\alpha_j\beta_j$ modulo x, $j = 1,2,3$, and writes $(s, \beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3)$; otherwise A puts $(\alpha_1, \alpha_2, \alpha_3)$ in the empty set E_s, $1 \leq s \leq 7$, with smallest index (i.e. $1 \leq j < s$ implies E_j not empty) and writes $(s, 0, 0, 0, 0, 0, 0)$.

6.6) If there is a triple $(\beta_1, \beta_2, \beta_3)$ in the set E_s, such that $\eta_{i_1}\beta_1, \eta_{i_2}\beta_2, \eta_{i_3}\beta_3$ are quadratic residues modulo x, A sets γ_j = a random square root of $\eta_{i_j}\beta_j$ modulo x, $j = 1,2,3$, and writes $(s, \beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3)$.

B's protocol.

1) B reads the collection $C = \{c_1, \ldots, c_m\}$ of clauses and sets E_0, E_1= empty set.

2) For $i = 1, \ldots, 7n$, let s_i the integer whose binary representation is $\sigma_{in-n} \cdots \sigma_{in-1}$. If $s_i \notin Z_x^{+1}$ then B discards s_i. Otherwise B reads the triple (d_1, d_2, d_3). If $d_2 \neq 0$ then

B checks that $s_i d_2 \equiv d_3^2 \bmod x$, that $d_2 \in E_{d_1}$ and puts s_i in E_{d_1}. If $d_2 = 0$ then B checks that E_{d_1} is empty and puts s_i in E_{d_1}.

3) B checks that E_0 and E_1 are not empty. B reads (y, y_0, y_1) and checks that $y = y_0 y_1 \bmod x$ and that $y_0 \in E_0$ and $y_1 \in E_1$.

4) B reads $(w_1, w_2, \ldots, w_k)$. B checks that $w_j \in Z_x^{+1}$, $j = 1, 2, \ldots, k$.

5) For each clause c_i, $i = 1, \ldots, m$, B performs steps 5.1-5.5.

5.1) B sets ST = empty stack and repeats n times step 5.2.

5.2) For $j = 0, \ldots, n-1$. Let z be the element whose binary representation is $\sigma_{7n^2+(i-1)n^2+(j-1)n} \cdots \sigma_{7n^2+(i-1)n^2+jn-1}$.
If $x \notin Z_x^{+1}$ then B discards z, otherwise B puts z in the stack ST.

5.3) B sets E_s = empty set, $s = 1, \ldots, 7$. B repeats step 5.4 until the number of elements in the stack ST is less than 3.

5.4) B picks three elements $\alpha_1, \alpha_2, \alpha_3$ from the stack ST. B reads the 7-tuple $(k_0, \ldots, k_6)$ from the letter. If $k_0 = 0$ then B checks that $\alpha_j = k_{j+3}^2 \bmod x$, $j = 1, 2, 3$. Otherwise if $k_0 = s$, $1 \le s \le 7$, and E_s is not empty B checks that $(k_1, k_2, k_3) \in E_s$ and that $k_j \alpha_j \equiv k_{j+3}^2 \bmod x$, $j = 1, 2, 3$, and then puts $(\alpha_1, \alpha_2, \alpha_3)$ in E_s. Otherwise B checks that $k_0 = s$, $1 \le s \le 7$, E_s is empty, and that E_j is not empty for $j = 1, \ldots, s-1$ and then puts $(\alpha_1, \alpha_2, \alpha_3)$ in E_s.

5.5) If each E_s, $s = 1, \ldots, 7$, is not empty then B reads the 7-tuple $(k_0, \ldots, k_6)$ from the letter. Let the clause c_i consists of literals $z_{i_1}, z_{i_2}, z_{i_3}$. If the literal z_{i_j}, $j = 1, 2, 3$, is the variable u_l then B sets $\eta_{i_j} = w_l$. If the literal z_{i_j}, $j = 1, 2, 3$, is the complement of the variable u_l then B sets $\eta_{i_j} = y w_l \bmod x$. B checks that $k_0 = s$, $1 \le s \le 7$, $(k_1, k_2, k_3) \in E_s$ and that $k_j \eta_{i_j} \equiv k_{j+3}^2 \bmod x$, $j = 1, 2, 3$.

If all the checks are successful B stops and accepts, otherwise B rejects.

In the following we prove Theorem 1 by showing that the above protocol is a Single-Theorem Non-Interactive ZKPS.

(A,B) satisfies the Completeness requirement.

Say that B has received a proof that the collection of clause $C = \{c_1 \ldots c_m\}$ over the set $U = \{u_1 \ldots u_k\}$ of boolean variables is satisfiable.

Assume that C is indeed satisfiable, let t be a truth assignment that satisfies C and suppose that A and B follow the specification of the protocol.

Let p_x, $x \in \mathcal{N}$, be the probability that a random y, $|\, y \,|\leq|\, x \,|$, is in Z_x^*, i.e. $p_x = Pr\big(y \leftarrow \{z|0 \leq z < 2^{|x|}\} : y \in Z_x^*\big)$. Notice that $p_x > 2/5$, for all $x \in Z_2(n)$ and sufficiently large n.

A halts at step 4 only when among the $7n$ integers obtained from σ those in Z_x^{+1} are all quadratic residues or all non quadratic residues modulo x. Denote with Q_x the probability of this event. One has that Q_x satisfies

$$Q_x \leq 2\Big(1 - \frac{p_x}{4}\Big)^{7n}, \tag{1}$$

and so is negligible.

Consider the relation $\overset{R}{=}$ on the set $Z_x^{+1} \times Z_x^{+1} \times Z_x^{+1}$ defined as:

$$(\alpha_1, \alpha_2, \alpha_3) \overset{R}{=} (\beta_1, \beta_2, \beta_3) \quad \text{iff} \quad (\alpha_i \beta_i, x) \in QR,\ i = 1, 2, 3.$$

It can be easily seen that $\overset{R}{=}$ is an equivalence relation and that the quotient set $(Z_x^{+1} \times Z_x^{+1} \times Z_x^{+1})/ \overset{R}{=}$ consists of 8 equivalence classes, $EQ_0, \ldots, EQ_7$. Let EQ_0 be the equivalence class formed of triple constituted of 3 quadratic residues.

If A follows the protocol properly then all the non empty sets E_s, $1 \leq s \leq 7$, are subsets of different equivalence classes. Furthermore all the triples that A does not put in any set E_s, $1 \leq s \leq 7$, at step 6.5 (that is the triples formed by 3 quadratic residues) are all in EQ_0. Therefore if A follows the protocol then he can always perform step 6.5 and all the checks made by B at step 1-5.5 are successful.

Consider now the clause c_i and let $\eta_{i_1}, \eta_{i_2}, \eta_{i_3}$ be the values computed at step 6.1 by A. Two cases are possible :

i) The sets E_s, $s = 1, \ldots, 7$ are non empty. Thus there is a set, E_r, such that each $(\beta_1, \beta_2, \beta_3) \in E_r$ is equivalent to $(\eta_{i_1}, \eta_{i_2}, \eta_{i_3})$. A can prove such an equivalence by simply showing $(\beta_1, \beta_2, \beta_3)$ and exhibiting random square roots modulo x of $\eta_{i_1}\beta_1, \eta_{i_2}\beta_2, \eta_{i_3}\beta_3$.

ii) Some of the E_s, $1 \leq s \leq 7$, are empty and $(\eta_{i_1}, \eta_{i_2}, \eta_{i_3})$ is not equivalent to any triple in the non empty sets.

From the above discussion we conclude that if A follows the protocol and doesn't halt at step 4, B always accepts. Therefore, the Completeness requirement of Single-Theorem Non-Interactive ZKPS's is met.

(A,B) satisfies the Soundness requirement.

B accepts a non satisfiable collection of clauses C either if

i) A cheated him during the proof that $x \in Z_2(n)$ and $(y, x) \in QNR$.

ii) $x \in Z_2(n)$ and $(y, x) \in QNR$ but A cheated him in proving that each clause is satisfied.

We now show that the probabilities of i) and ii) are negligible.

Suppose $x \in Z_s(n)$, $s > 2$. Consider the equivalence relation $\overset{\circ}{=}$ defined over Z_x^{+1} as

$$y_1 \overset{\circ}{=} y_0 \iff (y_1 y_2, x) \in QR.$$

Z_x^{+1} is partitioned by the relation $\overset{\circ}{=}$ in 2^{s-1} equivalence classes. The only case in which B accepts x is when among the $7n$ integers obtained from σ, those with Jacobi symbol $+1$ belong to at most two equivalence classes. Let P_s denote this probability. From

$$P_s \leq \binom{2^{s-1}}{2}\left(1 - \frac{2^{s-1} - 2^{s-2} - 1}{2^{s-1}} p_x\right)^{7n} + \binom{2^{s-1}}{1}\left(1 - \frac{2^s - 2^{s-1} - 1}{2^s} p_x\right)^{7n}$$

and $s < n$ we obtain that P_s is negligible.

The only case in which A can make B accept $(y, x) \notin QNR$ is when among the $7n$ integers obtained from σ those in Z_x^{+1} are all quadratic residues or all non quadratic residues modulo x. Note that when A chooses x, he already knows σ and so he could choose x such that this event occurs. So we have to show that the probability that, given $7n$ randomly chosen n-bit integers, $z_1, \ldots, z_{7n}$, there exists a $x \in Z_2(n)$ such that those integers $z_i \in Z_x^{+1}$ are all quadratic residues or all non quadratic residues modulo x is negligible. The probability, $Q(n)$, of this event is, from (1),

$$Q(n) \leq \sum_{x \in Z_2(n)} Q_x \leq 2^{n+1}\left(\frac{9}{10}\right)^{7n}.$$

Thus the probability of i) is negligible.

Now suppose that $x \in Z_2(n)$ and $(y, x) \in QNR$.

The only case in which ii) occurs is when there is an equivalence class, EQ_s, $1 \le s \le 7$, such that no triple of it is in the stack ST during the proof that a clause c_i is satisfied. Indeed if there is a triple for each EQ_s, $1 \le s \le 7$, then each E_s is not empty and if the clause c_i is not satisfied by the truth assignment t, A at step 6.6 cannot prove $(\eta_{i_1}, \eta_{i_2}, \eta_{i_3})$ to be equivalent to any triple in any of the sets E_s, $s = 1, ..., 7$.

On the other hand, if there is a EQ_s such that no triple of it is in the stack ST then A could split the set of triples formed of 3 quadratic residues in two parts. Then he could reveal that one of these two parts is formed of 3 quadratic residue by sending $(0, \ldots)$ along with the 3 square roots modulo x and finally could put the remaining triples in the empty set E_s. In this way A could show the equivalence of $(\eta_{i_1}, \eta_{i_2}, \eta_{i_3})$ with an element in such a set.

From the above discussion, we can say that the probability that B accepts a non satisfiable C, in the case ii), is no greater than the probability that during one of the n iterations of steps 6.1-6.5 at least one of the equivalence classes EQ_s, $s = 1, \ldots, 7$, has no triple in the stack ST. Let R denote this event. Then

$$\begin{aligned} Pr(R) &= \sum_{j=0}^{\lfloor n/3 \rfloor} Pr(R \mid \text{ there are } j \text{ triples in ST }) Pr(\text{there are } j \text{ triples in ST }) \\ &= \sum_{j=0}^{\lfloor n/3 \rfloor} \left(\frac{7}{8}\right)^j Pr(\text{ there are } j \text{ triples in ST }). \end{aligned}$$

The probability that there are l elements in the stack is $\binom{n}{l} q_x^l (1-q_x)^{n-l}$, where $q_x = p_x/2$ is the probability that at step 6.3 A puts the element z in the stack. Then the probability of putting j triples in the stack, for each clause c_i, is

$$\binom{n}{3j} q_x^{3j}(1-q_x)^{n-3j} + \binom{n}{3j+1} q_x^{3j+1}(1-q_x)^{n-3j-1} + \binom{n}{3j+2} q_x^{3j+2}(1-q_x)^{n-3j-2}.$$

Therefore

$$\begin{aligned} Pr(R) &\le \left(\frac{8}{7}\right)^{2/3} (1-q_x)^n \left(1 + \frac{\sqrt[3]{7}}{2}\frac{q_x}{1-q_x}\right)^{3\lfloor n/3 \rfloor + 2} \\ &= (1-q_x)^{n-3\lfloor n/3 \rfloor - 2} \left(\frac{8}{7}\right)^{2/3} \left(1 + \left(\frac{\sqrt[3]{7}}{2} - 1\right) q_x\right)^{3\lfloor n/3 \rfloor + 2} \\ &= O\big((1 - .02p_x)^n\big). \end{aligned}$$

(A,B) satisfies the Zero-Knowledge requirement.

For the Zero-Knowledge part, we exhibit, a Probabilistic Turing machine M that, in expected polynomial time, approximates the family of random variables $V = \{V(x)\}$, where $V(x) = \{\sigma \leftarrow \{0,1\}^{|x|^c}; y \leftarrow A(\sigma, x): (\sigma, y)\}$, over $3SAT$.

M's simulation protocol.

1) M chooses a random $x \in Z_2(n)$ along with its factorization, $x = pq$.
2) M sets $\sigma =$ empty string and repeats step 3 $7n$ times.
3) M chooses a random n-bit integer ν. If either $\nu \notin Z_x^{+1}$ then M adds the binary representation of ν to σ. Otherwise M chooses a random ρ in Z_x^* and adds the binary representation of $\rho^2 \bmod x$ to σ.
4) M adds n^3 random bits to σ. Let $\sigma = \sigma_0 \ldots \sigma_{7n^2+n^3-1}$.
5) M sets $E_0, E_1 =$ empty set.
6) For $i = 1, \ldots, 7n$, let s_i be the integer whose binary representation is $\sigma_{(i-1)n} \cdots \sigma_{in-1}$. If $s_i \notin Z_x^{+1}$ then M discards it. Otherwise M tosses a fair coin. If HEAD (TAIL) then M puts s_i in $E_0(E_1)$. If $E_0(E_1)$ is not empty M writes $(0, s, \alpha)\Big((1, s, \alpha)\Big)$, where s is a random element in $E_0(E_1)$ and α is a random square root modulo x of ss_i. If $E_0(E_1)$ is empty M writes $(0,0,\alpha)\Big((1,0,\alpha)\Big)$.
7) If E_0 or E_1 (or both) is empty then M halts.
   Otherwise, M randomly chooses $y_0 \in E_0$ and $y_1 \in E_1$, computes $y = y_0 y_1 \bmod x$ and writes (y, y_0, y_1).
8) M picks at random $r_j \in Z_x^*$, $j = 1, \ldots, k$, computes $w_j = r_j^2 \bmod x$ and writes $(w_1, w_2, \ldots, w_k)$.
9) From this point on, M performs the same protocol as A.
   (Note that M can perform A's protocol in polynomial time since he knows the factorization of x.)

The output of M is different from that of A only because in one case the string σ is random while in the other its first $7n^2$ bits are either the binary representation of elements not in Z_x^{+1} or quadratic residues modulo x. Under the quadratic residuosity assumption these two distributions are indistinguishable (which is not hard to prove).

3.2 A stronger version of our result.

The Single-Theorem Non-Interactive ZKPS of section 3.1 has a limited applicability. This is a drawback that is best illustrated by our conceptual example of the prover A who is leaving for his trip.

It is unlikely that for each theorem T that A finds, a string σ_T comes from the sky "devoted" to T and is presented to (is read by) both A and B. It is instead more probable, that A and B may have witnessed the same common random event of size n once, when and because they were together (or else, it is more probable that they generated a (common) random event. For instance by the coin flipping protocol as explained in section 1.1).

However the Proof System of section 3.1 will enable A to subsequently prove in Zero-Knowledge to B only a theorem of smaller size, roughly $\sqrt[3]{n}$ bit long. He is out of luck would he discover the proof of a theorem of bigger size.

Moreover, the n-bit long string A leaves with will not enable him to not interactively and in Zero-Knowledge prove to B many theorems. Below we modify the definition of Non-Interactive Zero-Knowledge Proof Systems with common coins and our solution so to allow A to prove to a B with which he shares an n-bit string, $poly(n)$ theorems of $poly(n)$ size.

We first define formally what this means.

Definition. *A Non-Interactive ZKPS is a pair (A,B) where A is a pair, (A_0, A_1), of Probabilistic Turing Machines and $B(\cdot,\cdot,\cdot,\cdot)$ is a deterministic algorithm running in time polynomial in the length of its first input, such that:*

1) **(Completeness)** *For all polynomials P, Q, and for all $(x_1, x_2, \ldots, x_{Q(n)}) \in (3SAT_{P(n)})^{Q(n)}$*

$$\begin{aligned} Pr\Big(\sigma \leftarrow \{0,1\}^{n^{O(1)}};\ & y_0 \leftarrow A_0(\sigma); \\ & y_1 \leftarrow A_1(\sigma, x_1, y_0); \\ & \vdots \qquad\qquad \vdots \\ & y_{Q(n)} \leftarrow A_1(\sigma, x_{Q(n)}, y_0) : \bigwedge_{j=1}^{Q(n)} B(x_j, y_j, y_0, \sigma) = 1\Big) > 1 - n^{-O(1)}. \end{aligned}$$

2) **(Soundness)** *For all polynomials P, Q, for all $(x_1, x_2, \ldots, x_{Q(n)}) \notin (3SAT_{P(n)})^{Q(n)}$*

and for each $A' = (A'_0, A'_1)$

$$\begin{aligned} Pr\Big(\sigma \leftarrow \{0,1\}^{n^{O(1)}};\ & y_0 \leftarrow A'_0(\sigma); \\ & y_1 \leftarrow A'_1(\sigma, x_1, y_0); \\ & \vdots \qquad\qquad \vdots \\ & y_{Q(n)} \leftarrow A'_1(\sigma, x_{Q(n)}, y_0) : \bigwedge_{j=1}^{Q(n)} B(x_j, y_j, y_0, \sigma) = 1\Big) < n^{-O(1)}. \end{aligned}$$

3) **(Zero-Knowledge)** *For each polynomial* Q*, the family of random variables* $V = \{V(x_1, \ldots, x_{Q(n)})\}$*, where*

$$\begin{aligned} V(x_1, \ldots, x_{Q(n)}) = \Big\{\sigma \leftarrow \{0,1\}^{n^{O(1)}};\ & y_0 \leftarrow A_0(\sigma); \\ & y_1 \leftarrow A_1(\sigma, x_1, y_0); \\ & \vdots \qquad\qquad \vdots \\ & y_{Q(n)} \leftarrow A_1(\sigma, x_{Q(n)}, y_0)\ : (\sigma, y_0, y_1, \ldots, y_{Q(n)})\Big\} \end{aligned}$$

is approximable over $\bigcup_n (3SAT)^{Q(n)}$.

Comment. Notice that we let A send first the proof of a theorem (y_0) other than the ones he is interested in communicating: $x_1, x_2, \ldots$. Notice that A is able to prove each "interesting" theorem essentially independently from all the other theorems (only memory of y_0 is needed by A to prove each single theorem). Also A need only to be poly-time if he has a suitable witness among his available inputs. This is not inherent to our definition of Non-Interactive ZKPS's. It is rather due to our specific solution. However it does not change the rules of the game in an essential way.

Notice also that B verifies each theorem independently (only memory of y_0 is needed); we essentially have one prover and many independent verifiers. Moreover, the proof of each subset of the interesting theorems is Zero-Knowledge.

Theorem 2. *Under the QRA, there exists a Non-Interactive ZKPS.*

Proof. The proof is similar to that of Theorem 1 but more technical and cumbersome. Our protocol uses random functions [GGM] to "expand" the initial random seed. It will appear in the final paper. □

3.3 Open Problems.

Our results can be extended in two directions.

The first extension deals with a scenario in which we have many independent provers, using the same random string σ to prove different theorems. A partial solution of this problem will appear in the final paper.

The second extension concerns the Complexity Theoretic Assumption on which our results are based. Namely, can we replace the QRA with the weaker assumption of the Existence of One-way functions? This question is discussed in the following and we address the reader to [DMP] for a partial solution of it.

3.3.1 Many Independent Provers.

We live in a scientific community in which all libraries possess copies of the same tables of random numbers prepeared by RAND corporation, the RAND tables. This is essentially a short string *shared* by the scientific community. Can we use the RAND tables to give one another Non-Interactive Zero-Knowledge Proofs?

Here the problem is not so much the fact that we share a random string of fixed length, rather than σ_n for each n. In fact the RAND table is long enough to allow us to prove an arbitrary polynomial number of theorems. The fact is that 3.2 tells us that a *single* prover releases Zero-Knowledge. Is this also true if we have (as it is the case) many provers? This problem is similar to that discussed in section 3.2.

We know that the answer is "Yes" if at most $O(\log n)$ provers are active, when a string of length n is available. The protocol can be accommodated to $M(n)$ provers. However each prover is obliged to invest a multiplicative factor of $M(n)$ in his computational effort (whether or not there really are $M(n)$ provers). This is unsatisfactory. It should be contrasted with the $P(\cdot)$ size and with the $Q(\cdot)$ many theorems of protocol 3.2.

We are thus naturally led to the definition of *Economical Non-Interactive ZKPS.*

Definition. *An Economical Non-Interactive ZKPS is a pair (A,B) where A is a Probabilistic Turing Machine and $B(\cdot,\cdot,\cdot)$ is a deterministic algorithm running in time polynomial in the length of its first input, such that:*

1) **(Completeness)** *For all polynomials P, Q, and for all* $(x_1, x_2, \ldots, x_{Q(n)}) \in$

$(3SAT_{P(n)})^{Q(n)}$

$$Pr\Big(\sigma \leftarrow \{0,1\}^{n^{O(1)}};\ y_1 \leftarrow A(\sigma, x_1);$$
$$\vdots \qquad \vdots$$
$$y_{Q(n)} \leftarrow A(\sigma, x_{Q(n)})\ :\ \bigwedge_{j=1}^{Q(n)} B(x_j, y_j, \sigma) = 1\Big) > 1 - n^{-O(1)}.$$

2) **(Soundness)** *For all polynomials P, Q, for all $(x_1, x_2, \ldots, x_{Q(n)}) \notin (3SAT_{P(n)})^{Q(n)}$ and for each A'*

$$Pr\Big(\sigma \leftarrow \{0,1\}^{n^{O(1)}};\ y_1 \leftarrow A'(\sigma, x_1);$$
$$\vdots \qquad \vdots$$
$$y_{Q(n)} \leftarrow A'(\sigma, x_{Q(n)}, y_0)\ :\ \bigwedge_{j=1}^{Q(n)} B(x_j, y_j, \sigma) = 1\Big) < n^{-O(1)}.$$

3) **(Zero-Knowledge)** *For each polynomial Q, the family of random variables $V = \{V(x_1, \ldots, x_{Q(n)})\}$, where*

$$V(x_1, \ldots, x_{Q(n)}) = \Big\{\sigma \leftarrow \{0,1\}^{n^{O(1)}};\ y_1 \leftarrow A(\sigma, x_1);$$
$$\vdots \qquad \vdots$$
$$y_{Q(n)} \leftarrow A(\sigma, x_{Q(n)})\ :(\sigma, y_1, \ldots, y_{Q(n)})\Big\}$$

is approximable over $\bigcup_n (3SAT)^{Q(n)}$.

Comment. Notice that the above definition is different from that of Non-Interactive ZKPS in the requirement that a proof of a theorem depends only on σ and not on any previously proved theorem (y_0).

3.3.2 Relaxing the assumption.

Our protocol relies on the fact that deciding Quadratic Residuosity is *hard.*

One would like to prove our result under the assumption that one-way functions exist. This is the weakest possible assumption in Cryptography, since if one-way functions do not exist then public-key cryptography is not possible. In [DMP] we present a partial solution to this problem. Namely we exhibit a protocol that allows a prover to non interactively prove any theorem of size n after an interactive preprocessing step whose computational effort is roughly n^3.

References.

[B] M. Blum, *Coin Flipping By Telephone*, IEEE COMPCON '82.

[BC] G. Brassard, C. Crepeau, *Non Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for Sat and Beyond*, Proceedings of the 27th Symposium on Foundations of Computer Science, 1986.

[BFM] M. Blum, P. Feldman, S. Micali, in preparation.

[DMP] A. De Santis, S. Micali, G. Persiano, in preparation.

[FMRW] M. Fischer, S. Micali, C. Rackoff and D. Witenberg, *A Secure Protocol for the Oblivious Transfer*, in preparation 1986.

[FFS] U. Feige, A. Fiat and A. Shamir, *Zero Knowledge Proofs of Identity*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987.

[GGM] O. Goldreich, S. Goldwasser, S. Micali, *How to Construct Random Functions*, Journal of ACM, vol. 33, No. 4, October 1986.

[GHY] Z. Galil, S. Haber, M. Yung, *A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystem*, Proceedings of the 26th Symposium on Foundations of Computer-Science, 1985.

[GJ] M. Garey, D. Johnson, *Computers and Intractability : a Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, 1979.

[GM] S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Science, vol. 28, No. 2, 1984.

[GMR] S. Goldwasser, S. Micali, C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, Proceedings of the 17th Annual ACM Symposium on Theory of Computing, 1985.

[GMW1] O. Goldreich, S. Micali, A. Wigderson, *Proofs That Yield Nothing But Their Validity and a Methodology of Cryptographic Protocols Design,* Proceedings of the 27th Symposium on Foundations of Computer-Science, 1986.

[GMW2] O. Goldreich, S. Micali, A. Wigderson, *How to Play Any Mental Game,* Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987.

[GoMiRi] S. Goldwasser, S. Micali, R. Rivest, *A Digital Signature Scheme Secure Against Adaptive, Chosen Cyphertext Attack,* to appear in SIAM J. on Computing.

How to Solve any Protocol Problem - An Efficiency Improvement

(Extended Abstract)

Oded Goldreich   *Ronen Vainish*

Dept. of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

Consider n parties having local inputs $x_1, x_2, ..., x_n$ respectively, and wishing to compute the value $f(x_1, \ldots, x_n)$, where f is a predetermined function. Loosely speaking, an n-party protocol for this purpose has *maximum privacy* if whatever a subset of the users can efficiently compute when participating in the protocol, they can also compute from their local inputs and the value $f(x_1, \ldots, x_n)$.

Recently, Goldreich, Micali and Wigderson have presented a polynomial-time algorithm that, given a Turing machine for computing the function f, outputs an n-party protocol with maximum privacy for distributively computing $f(x_1, \ldots, x_n)$. The maximum privacy protocol output uses as a subprotocol a maximum privacy two-party protocol for computing a particular simple function $p_1(\cdot,\cdot)$. More recently, Haber and Micali have improved the efficiency of the above n-party protocols, using a maximum privacy two-party protocol for computing another particular function $p_2(\cdot,\cdot)$. Both works use a *general* result of Yao in order to implement protocols for the *particular* functions p_1 and p_2.

In this paper, we present direct solutions to the above two particular protocol problems, avoiding the use of Yao's general result. In fact, we present two alternative approaches for solving both problems. The first approach consists of a simple reduction of these two problems to a variant of *Oblivious Transfer*. The second approach consists of designing direct solutions to these two problems, assuming the intractability of the Quadratic Residuosity problem. Both approaches yield simpler and more efficient solutions than the ones obtained by Yao's result.

This research was supported by grant No. 86-00301 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

1. INTRODUCTION

The main purpose of many cryptographic protocols is to allow parties to collaborate towards some common goal, while maintaining the maximum possible privacy of their secrets. Typically, the common goal is to compute some function of the local inputs (secrets) held by the different parties. Maximum privacy means that this value is distributively computed without revealing more about the local inputs than what is revealed by the value itself.

More formally, let x_i be the local input of party i $(1 \le i \le n)$, and f be an n-argument function. The parties wish to obtain the value $f(x_1, \ldots, x_n)$, but do not wish to leak any further information about their local inputs. To better understand what is meant by this requirement, consider the situation when all parties trust an additional party. In this case, each party may (secretly) send his local input to the *trusted party*, which will then compute the value of the function, and announce this value to all parties. A maximum privacy protocol achieves the effect of the trusted party without using a trusted party. Namely, whatever a party can efficiently compute when participating in a maximum privacy protocol, he could have efficiently computed after participating in the above "trusted party" protocol.

To better understand what is meant by maximum privacy, consider the problem of computing the sum of the local inputs (i.e. $f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i$). A maximum privacy protocol for computing the sum of the local inputs guarantees that whatever a coalition T of parties can efficiently compute when participating in the protocol, can also be efficiently computed from their local inputs $(\{x_i\}_{i \in T})$ and the sum of all local inputs (i.e. $\sum_{i=1}^{n} x_i$). Equivalently, all they learned about the local inputs of the other parties is their sum (i.e. $\sum_{i \notin T} x_i$), and this of course can not be avoided.

Goldreich, Micali and Wigderson [GMW2] have proposed a method for generating maximum privacy protocols for computing any function f. Their method is in fact a polynomial-time algorithm that given as input a Turing Machine description of the (n-argument) function f, outputs a maximum privacy n-party protocol for computing f. These protocols use instances of a maximum privacy two-party subprotocol for the following particular protocol problem (in S_5 - the symmetric group of 5 elements):

Input: A's local input is a permutation, $\tau \in S_5$, while B's local input is a permutation $\sigma \in S_5$.
Output: A's local output is a permutation $\tau' \in S_5$, and B's local output is a permutation $\sigma' \in S_5$ such that $\tau \cdot \sigma = \sigma' \cdot \tau'$. (Here $\cdot$ means permutation composition.)

Recently, an efficiency improvement of the [GMW2] algorithm has been suggested by Haber and Micali [HM]. The protocols output by their algorithm use instances of a maximum privacy two-party subprotocol for the following particular problem (in $GF(2)$ arithmetic):

Input: A's local input is a pair of bits a_1 and a_2, while B's input consists of the bits b_1 and b_2.
Output: A's local output is a bit a_0, while B's local output is a bit b_0 such that $a_0 + b_0 = \sum_{i=1}^{2} a_i \cdot b_i$.

Both, [GMW2] and [HM] use for solving the above problems, a general result of Yao [Y2]: a method for generating maximum privacy two-party protocols for any two-argument function. This method (modified in [GMW2] by using ideas from [EGL]) guarantees maximum privacy under the assumption that trapdoor one-way permutations exist.

In this paper, we present direct solutions to the above two particular protocol problems, avoiding the use of Yao's general result. In fact, we present two alternative approaches for solving both problems. The first approach consists of a simple reduction of these two problems to a variant of *Oblivious Transfer*, which can implemented assuming the existence of any trapdoor one-way permutation. The second approach consists of designing direct solutions to these two problems, assuming the intractability of the Quadratic Residuosity problem. Both approaches yield simpler and more efficient solutions than the ones obtained by Yao's result.

Our protocols and their applications are presented in a model where parties follow the protocol properly, except that they may store all intermediate computations done during the execution. Thus, we concentrate only in guaranteeing that the protocols have maximum privacy. Using the results of [GMW2], each maximum privacy protocol in the above model can be transformed into a protocol guaranteeing both maximum privacy and correctness of output in a model where a minority of the parties may deviate from the protocol in arbitrary (but polynomial-time) manner.

2. PRELIMINARIES

In this section, we recall the basic definitions and notatiôns used in this paper.

2.1. Two-party Cryptographic Protocols

Loosely speaking, a two-party cryptographic protocol is a pair of programs run by a corresponding pair of interacting Turing Machines. An *interactive Turing Machine* is a Turing machine with the following tapes:

1) A read only *input tape*.
2) A read only *random tape*.
3) A read/write *working tape*.
4) A pair of communication tapes, one being read-only and the other write-only.
5) A write only *output tape*.

The machine can be thought of as using the bits of the random tape as coin tosses, sending messages through its write-only communication tape, and receiving messages on its read-only communication tape. It should be noted that the current configuration of the machine is determined by the context of its input tape, random tape, and read-only communication tape.

Two interactive machines A and B, are called an *interactive pair* (of Turing machines) if they share their communication tapes in the obvious manner (i.e. A's read-only communication tape is B's write-only communication tape and via versa).

We consider only polynomial-time protocols. These are protocols consisting of pairs of programs, such that the running time of each program is polynomial in the length of the input.

Typically, we will not be interested in a particular execution of a protocol, but rather in the probability distribution on the set of possible executions. This probability distribution is a function of the local inputs and internal coin tosses of the interacting machines.

A particularly intersting probability distribution is defined by a party's view of the execution. The party's view of an execution contains the contents of his local input and random tapes, as well as the contents of his read-only communication tape. We stress that the contents of the input tape and the read-only communication tape can not be modified (or erased) by the program. We denote by $A_{B(y)}(x)$ the probability distribution defined by A 's view of an execution, in which A has local input x, and B has local input y.

2.2. Polynomial Indistinguishability

A fundamental notion regarding probability distributions is the inability to efficiently tell them apart. This notion is captured by the definition of polynomially indistinguishable probability ensembles originating in [GM, Y1] and sketched below.

A probability ensemble $Y=(Y_1,Y_2,...)$ is an infinite sequence of probability distributions, where Y_k is a probability distribution on binary strings. Typically, the support of Y_k will contain strings of length polynomial in k. A *test*, T, is a probabilistic polynomial time algorithm that on input a string x output a bit b. Let $P_{Y_k}^T$ denote the probability that T outputs 1 on input a string randomly selected with a probability distribution Y_k. Two ensembles X and X' are *polynomially indistinguishable* if for all tests T, for all constants $c>0$, and for sufficiently large k,

$$|P_{X_k}^T - P_{X'_k}^T| < \frac{1}{k^c}$$

2.3. The Privacy Requirement

In the introduction, we have motivated maximum privacy protocols as ones allowing the distributed computation of functions without revealing more about the local inputs than what is revealed by the value of the function. It was required that whatever a party can efficiently compute when participating in a maximum privacy protocol, he could have efficiently computed from his local input and the value of the function. Clearly, it suffices to require that he can efficiently compute his view of the execution from his local input and the value of the function. A formal definition follows. (For first reading of the definition, assume that z is the integer k in unary representation and that its sole purpose is to allow the used of the formalism of polynomial indistinguishability.)

Definition 1 (A program preserves privacy with respect to a particular program.): Let Π be a probability ensemble $(\Pi_1,\Pi_2,...)$ such that Π_k is a probability distribution on triples (x,y,z). Program B *preserves the privacy of* f *with respect to program* A if there is a probabilistic polynomial-time machine M, that for every ensemble Π, when given input x,z and $f(x,y)$ outputs $M(x,z,f(x,y))$ such that ensemble $M(x,z,f(x,y))$ is polynomially indistinguishable from the ensemble $A_{B(y)}(x,z)$. (Here the triple (x,y,z) is chosen with probability distribution Π_k.)

In fact, we allow z to be arbitrary, thus capturing a priori information that party A might have had on the input of B. This way, we guarantee that even with the help of such a priori information, executing the protocol does not reveal more about the local inputs than is revealed by the value of $f(x,y)$.

Although maximum privacy is defined here with respect to the computation of functions, the definition naturally extends to the computation of probability distributions.

2.4 Two Models of Party's Behavior

In this paper we consider two types of party's behavior. The first type, called *semi-honest behavior* consists of a party following his program while recording all intermediate computing steps on a special tape (called the *history tape*) and conducting an arbitrary polynomial-time computation using the history tape as an input. Note that even if a program of a semi-honest specifies that it has to erase the contents of his working tape, this contents still appear on the history tape. Yao [Y2] (resp. Goldreich, Micali and Wigderson [GMW2]) presents a method of "forcing" the participants of any two-party (resp. multi-party) protocol to behave in a semi-honest manner.
Definition 2 (Protocol which preserves privacy in the semi-honest model): A two-party protocol (A,B) *preserves the privacy of f in the semi honest model* if program A preserves privacy with respect to program B and program B preserves privacy with respect to program A.
Definition 3 (Maximum privacy protocol.): A protocol (A,B) has *maximum privacy* if program A preserves privacy with respect to all polynomial-time program B^* and program B preserves privacy with respect to all polynomial-time program A^*.

When talking about a cryptographic protocol we are usually interesting in two properties, correctness and privacy (correctness means that the true value of f is being computing by the protocol.). In this paper we are concerned only with the privacy condition. This can be motivated in two ways. First we believe that privacy and correctness are distinct notions which are better understood when dealt separately. Correctness is easily dealt using zero-knowledge proofs [GMW1], while privacy even in the semi-honest model requires different techniques [GMW2]. Secondly, it is natural to consider setting in which the parties are very interested in obtaining the correct value of the function and on top of this seek to gain additional information (but not at the cost of not getting the correct value). This is formulated by the following behavior model.

A *value-preserving adversary*, consists of a party which may deviates from the protocol in any manner that does not change the true value of f.

We introduce two protocols for the same problem, the first preserve privacy in the semi-honest model while the second has maximum privacy.

2.5 One-out-of-Two Oblivious Transfer

Susan and Ron are friends. Susan has two secret bit, which Ron wants. In order to preserve their friendship Susan is willing to give Ron only one of her secret at his choice, but Ron does not want her to know which secret he chose. A *one-out-of-two Oblivious Transfer*, denoted OT_2^1, is a two-party protocol which guarantees that Ron gets only the secret (bit) he has chosen while Susan does not know which secret he chose.

The OT_2^1 as motivated above, must be related to a model of behavior. We consider OT_2^1 in the semi-honest model and in the value-preserving adversary model. When we say that a protocol implements OT_2^1 in a specific model we mean that the OT_2^1 properties hold when the party's behavior is restricted to is model.

2.6. The Quadratic Residuosity Problem

Let m be a composite integer, the product of two large primes p and q. We denote by Z_m^* the multiplicative group modulo m. The set of quadratic residue modulo m is denoted by

$$Q_m=\{a: \exists x \in Z_m^* \ s.t.\ a \equiv x^2\,(m)\}$$

For every $a \in Z_m^*$, the Jacobi symbol of a mod m, denoted $(\frac{a}{m})$, is defined as $(\frac{a}{p})\cdot(\frac{a}{q})$, where $(\frac{a}{p})$ is $+1$ if a is a quadratic residue modulo p and -1 otherwise. The Jacobi symbol $(\frac{a}{m})$ can be easily computed from a and m. Clearly, $(\frac{a}{m})=-1$ implies $a \notin Q_m$, but the converse does not hold. In fact, distinguishing elements of Q_m from quadratic non-residues mod m (with Jacobi Symbol 1) is considered intractable. (This computation is easy if the factorization of m is known.) To concentrate on elements with Jacobi Symbol 1, we denote

$$Z_m^{(+1)}=\{a \in Z_m^*: (\frac{a}{m})=+1\}.$$

$$N_m=Z_m^{(+1)}-Q_m$$

The *Quadratic Character* of x mod m, denoted $QC_m(x)$, is defined as 0 if $x \in Q_m$ and 1 otherwise. The *Quadratic Residuosity problem* is to determine, on input x and m, the value of $QC_m(x)$. This task is considered intractable in the following sense

Intractability Assumption of Quadratic Residuosity [GM]: Let $C=\{C_i\}$ be an infinite sequence of Boolean circuits such that C_i has $2i$ input bits. Let f_{C_i} denote the fraction of integers m product of two primes, of length $i/2$ bits each, such that for every $x \in Z_m^{(+1)}$, $C_i(x,m)=QC_m(x)$. For every family of polynomial size circuits, $C=\{C_i\}$, every constant $c>0$ and sufficiently large i, $f_{C_i}<i^{-c}$. (Here the size of a circuit family $C=\{C_i\}$ is a function mapping i to the number of gates in C_i.)

We use the fact that, under the intractability assumption of Quadratic Residuosity, it is infeasible to guess the quadratic character with any non-negligible advantage over 1/2.

Definition: We say that C *polynomially approximates* Quadratic Residuosity, if there exist a constant $c>0$ such that for infinitely many i's

$$Prob\left[C_i(x,m)=QC_m(x)\right] > \frac{1}{2}+\frac{1}{n^c}$$

where the probability is taken over all possible $m=p\cdot q$ (with p and q being two primes of length $i/2$ each) and all $x \in Z_m^{(+1)}$ with uniform probability distribution.

Theorem [GM]: Under the intracability Assumption of Quadratic Residuosity, there exist no family of polynomial size circuits that polynomially approximates Quadratic Residuosity.

2.7. Notations

Let S be a finite set. By $e \in_R S$ we mean an element randomly chosen from the set S with uniform probability distribution.

We denote by S_5 the group formed by the set of all permutations over $\{1,2,3,4,5\}$, and permutation composition as operator. (This group is known as the symmetric group.)

3. THE GF(2) SCALAR PRODUCT PROTOCOL

In this section we present a maximum privacy two-party protocol for the problem of distributively computing scalar product in $GF(2)$, defined as follows:

Input: A's local input is a t-dimensional binary vector $\bar{a}=(a_1,a_2,\ldots,a_t)$, while B's input is another t-dimensional binary vector $\bar{b}=(b_1,b_2,\ldots,b_t)$.

Output: A's local output is a bit a_0, while B's local output is a bit b_0 such that $a_0+b_0=\sum_{i=1}^{t} a_i\cdot b_i$.

In fact, we are interested in the case $t=2$, which is exactly the subprotocol required for the Haber Micali protocol generator [HM]. For simplicity, we present a protocol for the following related problem:

Input: A's local input is a pair of bits a_0 and a_1, while B's input is a single bit b_1.

Output: B's local output is a bit b_0 which equals $a_0+a_1\cdot b_1$. (A has no local output.)

It is easy to reduce the original problem to the later problem. Alternatively, one may use the ideas of the protocol described below to directly solve the original problem.

3.1 Protocol For Semi-Honest Using OT_2^1.

A defines its first secret to be a_0 and his second secret to be a_0+a_1. Using OT_2^1 B chooses one of A's secrets according to the value of b_1. If $b_1=0$ then B chooses the first secret, otherwise he chooses the second secret.

It easy to see that B's output bit equals $a_0+a_1b_1$, thus correctness holds. The privacy in the semi-honest model holds by the definition of OT_2^1 in the semi-honest model.

An OT_2^1 is simply implemented in the semi-honest model, assuming the existence of trapdoor one way permutation [GMW2], unfortunately this implementation does not have maximum privacy. Zero knowledge proofs can be used in order to ensure that this OT_2^1 protocol has maximum privacy, however the modified protocol is no longer simple and efficient.

An efficiency improvement have been achieved in the next maximum privacy protocol for the scalar product problem, that is under the Quadratic Residuosity Assumption.

3.2. The Protocol in the Value-Preserving Adversary model.

Preprocessing: B chooses at random two k-bit primes p, q. (k is the security parameter) B computes $m=p\cdot q$. Next, B chooses $y \in_R N_m$ and publishes the couple m, y.

i) B chooses $s \in_R Z_m^*$ and computes $\beta=(s^2 \cdot y^{b_1} \bmod m)$. B sends β to A.

ii) A chooses $r \in_R Z_m^*$ and computes $\alpha=(r^2 \cdot y^{a_0} \cdot \beta^{a_1} \bmod m)$. A sends α to B.

iii) B checks the quadratic residuosity of α, and sets $b_0=QC_m(\alpha)$.

3.3. Correctness of the Protocol

We first show, that the above protocol is correct; namely that the output satisfies the specification conditions.

Claim 1: The bit b_0 computed by B does equal $a_0+a_1 \cdot b_1$.

Proof: B gets

$$\alpha \equiv r^2 \cdot y^{a_0} \cdot \beta^{a_1} \equiv r^2 \cdot y^{a_0} (s^2 \cdot y^{b_1})^{a_1} \equiv (r \cdot s^{a_1})^2 \cdot y^{a_0+a_1 b_1} \pmod{m}.$$

B set $b_0 = QC_m(\alpha) \equiv a_0+a_1 \cdot b_1 \pmod 2$. □

3.4. Maximum Privacy of the Protocol

We now prove that the above protocol has the maximum privacy property. First we use the Intractability Assumption of subsection 2.6 to prove that B preserves privacy with respect to any A^*, and next we prove that A preserves privacy with respect to any B^* (using no assumptions).

By Definition 1, program B preserves privacy with respect to A^* if there exists a machine which, on input the local inputs of A^* and the value of the function, simulates the interaction between A^* and B. This requirement has to be satisfied for any possible input that A^* may have, including encoding of possible a-priori information on B's inputs (denoted z). However, if the modulus m is chosen in the preprocessing and is input to the protocol then z may depend on it. In particular, z may contain the prime factorization of m and in such a case clearly B does not preserve privacy. This problem may be resolved in one of the following ways:

1) Having B choose m at random each time the protocol is executed, instead of having it chosen in a preprocessing stage. This completely solves the problem, at the cost of substantially decreasing the efficiency of the protocol.

2) Leaving the protocol as it is, and relaxing the definition of privacy preserving. The definition is relaxed by restricting z to be polynomial-time computable. In particular, $z=R(b_1, m, y)$, where R is a probabilistic polynomial-time algorithm. (Thus, z may be a random variable.) This restriction is justified by the applications of the above protocol.

Typically, the protocol will be used many times, each time with the same modulus (m) but with possibly different a_0,a_1,b_1. When considering the i-th application of the protocol, the input to A^* is the history of the previous $i-1$ applications. One can then use induction on i to show that privacy is preseved in i successive applications of the protocol. In the induction step, we use the fact that the history of the previous $i-1$ can be simulated by a probabilistic polynomial-time machine, and thus the input z in the current application satisfies the restriction.

In the following Claim, we use adopt the second alternative. The reader may easily modify our proof to show that the modified protocol (as suggested in the first alternative) preserves privacy in the original sense (of Definition 1).

Claim 2: Assuming intractability of Quadratic Residuosity and restricting z to be polynomial-time computable (see (2) above), program B of the above protocol preserves privacy with respect to any A^*.

Proof's Sketch: To prove the claim, we demonstrate a machine M which on input a_0,a_1, m,y and z (as restricted above) outputs a probability distribution $M(a_0,a_1,m,y,z)$ which is polynomially indistinguishable from $A^*_{B(b_1,p,q,m,y)}(a_0,a_1,m,y,z)$. Machine M proceeds as follows:

(Simulates step (i) of machine B): Sets $b'_1=1$, chooses $s \in_R Z^*_m$, and computes $\beta'=(s^2 \cdot y^{b'_1} \bmod m)$. Outputs its inputs together with β' and stops.

We will show that the output of M is polynomially indistinguishable from the contents of the input and read-only communication tapes of A^* (when interacting with B). The only potential difference between $M(a_0,a_1,m,y,z)$ and $A^*_{B(b_1,p,q,m,y)}(a_0,a_1,m,y,z)$ may be created by a difference between the distribution of β and β'.

There are essentially two cases.

Case 1: $Prob(b_1=1)>1-k^{-c}$, for all $c>0$ and sufficiently large k. In such a case, the ensembles $M(\cdots)$ and $A_{B(\ldots)}(\cdots)$ are almost the same and can not be polynomially distinguished (regardless of the difficulty of determining Quadratic Residuosity).

Case 2: There exist a constant $c>0$ such that $Prob(b_1=0)>k^{-c}$ for infinitely many k's. Assume, on the contrary, that there is a (polynomial-time) test T distinguishing $M(a_0,a_1,m,y,z)$ from $A^*_{B(b_1,p,q,m,y)}(a_0,a_1,m,y,z)$, when (a_0,a_1,m,y), (b_1,p,q,m,y) and z are taken from a distribution, denoted Π_k, in which p,q are randomly selected k-bit primes, $m=pq$, $y \in_R N_m$, and $z=R(b_1,m,y)$, where R is a probabilistic polynomial-time machine. In such a case, we use the test T to construct a family of circuits for approximating Quadratic Residuosity. Let I_k be a value of (a_0,a_1) for which the test T distinguishes the above two ensembles. With no loss of generality, assume that T outputs 1 with higher probability on the ensemble $M(\cdots)$ than on the ensemble $A_{B(\ldots)}(\cdots)$.

The k-th circuit incorporates I_k and the test T, working as follows: On input a k-bit composite m and $x \in Z_m^{(+1)}$, the circuit computes y and z as explained below, feeds T with (I_k,m,y,z,x) and outputs T's answer. (x is placed in the position of β (β').) It is left to specify the computation of y and z.

The circuit chooses $y \in_R Z_m^{(+1)}$, and computes $z=R(b_1,m,y)$ using the probabilistic polynomial-

time machine R (which is incorporated in the circuit). Note that the test T will determine correctly the quadratic character of $x \in Z_m^{(+1)}$, in case $y \in N_m$. We do not know how T behaves in case $y \in Q_m$. Therefore, before using y to test the quadratic character of x we estimate the behavior of the test with this y. Namely, we select many $r_i \in_R Q_m$ (by letting $r_i = s_i^2$, where $s_i \in_R Z_m^*$) and feed the test T with either (I_k, m, y, z, r_i) or $(I_k, m, y, z, r_i \cdot y)$. If the test T distinguishes these two cases, we use this y for determining the quadratic character of x (i.e. feed T with (I_k, m, y, z, x)). Otherwise, we try again.
One can show that the circuits constructed as above do approximate Quadratic Residuosity with a non-negligible advantage. The technical details are quite standard, and are omitted here. We reach a contradiction to the Quadratic Residuosity Assumption, and the Claim follows. □

We now prove preservation of privacy with respect to B. This time we use no intractability assumptions.

Claim 3: The protocol preserves privacy with respect to B.
Proof's Sketch: We will show that there exist a machine M which on input b_1, k, z and $f(a_0, a_1, b_1)$ $(= a_1 \cdot b_1 + a_0 \bmod 2)$, outputs a probability distribution which is identical to the distribution on B's input and read-only tapes during interaction with A. M operates as follows.

1) M randomly chooses two k-bit primes p, q, computes $m = p \cdot q$, and chooses $y \in_R (Z_m^{(+1)} - Q_m)$.
2) M chooses $r' \in_R Z_m^*$, computes $\alpha' \equiv r'^2 \cdot y^{f(a_0, a_1, b_1)} \pmod m$, and outputs (its input and) α'.

Recall that A calculate α as $\alpha \equiv r^2 \cdot y^{a_0} \cdot \beta^{a_1} \equiv (r \cdot s^{a_1})^2 \cdot y^{a_0 + a_1 b_1} \pmod m$, where $r \in_R Z_m^*$. Since both $r \cdot s^{a_1}$ and r' are uniformly distributed in Z_m^*, α and α' have identical probability distribution. The Claim follows. □

3.5. Summing Up

Combining Claims 1, 2 and 3, we get

Theorem 1: The above protocol is a maximum privacy protocol for the simplified $GF(2)$ scalar product problem.

A protocol for the original $GF(2)$ scalar product problem, can be easily derived and proven using the above ideas. The protocol of subsection 3.1 is modified as follows. In step (i), B chooses $s_1, s_2, \ldots, s_t \in_R Z_m^*$, computes $\beta_i = (s_i^2 \cdot y^{b_i} \bmod m)$, and sends $\beta_1, \beta_2, \ldots, \beta_t$ to A. In step (ii), A computes $\alpha_i = \beta_i^{a_i}$, chooses $r \in_R Z_m^*$ and $a_0 \in_R \{0,1\}$, computes $\alpha = (r^2 \cdot y^{a_0} \cdot \Pi_{i=1}^t \alpha_i \bmod m)$, and sends α to B. In step (iii), B sets $b_0 = QC_m(\alpha)$ $(= a_0 + \sum_{i=1}^{t} a_i b_i)$.

4. THE PERMUTATION SWITCHING PROTOCOL

In this section, we present a two-party protocol with maximum degree privacy for the problem of switching permutations defined as follows:

Input: A's local input is a permutation, $\tau \in S_5$, while B's local input is a permutation $\sigma \in S_5$.
Output: A's local output is a permutation $\tau' \in S_5$, and B's local output is a permutation $\sigma' \in S_5$ such that $\tau \cdot \sigma = \sigma' \cdot \tau'$.

An equivalent formulation used in the sequel is
Input: A's local input is a pair of permutations, $\tau, \tau' \in S_5$. B's local input is a permutation $\sigma \in S_5$.
Output: B's local output is $\sigma' = \tau \cdot \sigma \cdot \tau'$. ($A$ has no local output.)

One-out-of 120 Oblivious Transfer can be implements to solve the above problem in the semi-honest model. A maximum privacy protocol for the permutation switching problem is following presents, that is under the Quadratic Residuosity Assumption.

4.0. Conventions

Throughout this section, we use quite non-standard representation of permutations. The reason for this representation is that it allows to composite a non-encrypted permutation with an encrypted permutation resulting in an encrypted permutation.

We represent permutations in S_5 by quintuples of distinct elements in $\{1,2,3,4,5\}$. By $(i_1, i_2 ... i_5)$ we mean the permutation mapping i_k to k ($\forall k = 1...5$). For example let $\sigma = (3,5,1,2,4)$, then $\sigma \cdot (A, B, C, D, E) = (C, E, A, B, D)$.
Assume that we encrypt quintuples $(i_1, \ldots, i_5)$ by encrypting each element separately; namely $E(i_1, \ldots, i_5) = E(i_1), \ldots, E(i_5)$. Then, given $\sigma = (3,5,1,2,4)$ and $E(i_1, \ldots, i_5)$ we can compute

$$E(\sigma \cdot (i_1, i_2, i_3, i_4, i_5)) = E(i_3, i_5, i_1, i_2, i_4) = E(i_3), E(i_5), E(i_1), E(i_2), E(i_4) = \sigma \cdot E(i_1, \ldots, i_5)$$

The representation used above allows us to compute $E(\tau \cdot \sigma)$ from τ and $E(\sigma)$. We wish to be able to compute $E(\tau \cdot \sigma)$ from $E(\tau)$ and σ. Using two particular encryption formats, E and $\bar{E}$, we are able to compute $\bar{E}(\tau \cdot \sigma)$ from $E(\tau)$ and σ.

We encrypt quintuples $\sigma = (i_1, \ldots, i_5)$ by encrypting each element separately. Namely $E(\sigma) = E(i_1), \ldots, E(i_5)$. To encrypt an element $i \in \{1,2,\ldots,5\}$ we use a quintuple of elements in $Z_m^{(+1)}$ ($m = p \cdot q$ is composed of two large primes), with a quadratic residue in the i-th location and quadratic non-residues in all other locations. Specifically $E_m(i)$ is a probabilistic encryption equaling $(s_1, s_2 ... s_5)$, where $s_i \in_R Q_m$ and $s_j \in_R N_m$, for all $j \neq i$. For notational convenience we use $E(\cdot)$ instead of $E_m(\cdot)$. For simplicity, we use the shorthand (Q, N, N, N, N) for $E(1)$ ((N, Q, N, N, N) for $E(2)$, etc.), where Q denotes $s \in_R Q_m$ and N denotes $s \in_R (Z_m^{(+1)} - Q_m)$.

The advantage of this encryption method is that it allows us to compute an encryption of the Boolean predicate $i=j$ from $E(i)$ and $E(j)$, without yielding any additional information about $i, j \in \{1,2,3,4,5\}$. Given $E(i)$ and $E(j)$, we first apply coordinate-wise multiplication to the two quintuples, and next apply a random permutation to the result. In case $i=j$, coordinate-wise multiplication yields a quintuple of Quadratic residues. In case $i \neq j$, coordinate-wise multiplication yields a quintuple with Quadratic Non-residues in the i-th and j-th location, and Quadratic residues elsewhere. For example, coordinate-wise multiplication of $E(2) = (N, Q, N, N, N)$ by $E(4) = (N, N, N, Q, N)$ yields (Q, N, Q, N, Q). Applying a random permutation to the result, yields (in case $i \neq j$) a quintuple with two Quadratic Non-residues.

4.1. The Protocol in the Value-Preserving Adversary model.

Preprocessing: B chooses at random two k-bit primes p, q. (k is the security parameter)
B computes $m = p \cdot q$. Next, B chooses $y \in_R N_m$ and publishes the couple m, y.

1) B encrypt (his input) σ, using Quadratic non-residues and residues mod m. (Encryption is as specified above.) B sends $E(\sigma)$ to A.

2) A computes $E(\tau \cdot \sigma)$ by applying (his first input) τ to $E(\sigma)$, as described above.

Let $(i_1, i_2, \ldots, i_5) = \tau\sigma\tau'$, $(\tau'_1, \tau'_2, \ldots, \tau'_5) = \tau'$, and $(e_1, e_2, \ldots, e_5) = E(\tau\sigma)$.
for $j=1$ to 5 do begin (steps $3.j$ and $4.j$):

3.j) A computes $\vec{E}(i_j)$ as follows. First, A picks new probabilistic encryptions $E(1), E(2), \ldots, E(5)$ (using m and y).
for $l=1$ to 5 A computes the coordinate-wise multiplication of e_j and $E(l)$, and randomly permutes the resulting quintuple. Denote the randomly permuted result by $r_{j,l}$.
A forms five pairs $(r_{j,1}, \tau'_1), (r_{j,2}, \tau'_2), \ldots, (r_{j,5}, \tau'_5)$, orders the pairs by their rightmost element, and sends the pairs (in this order) to B.

4.j) B retrieves i_j as follows. Among the five pairs received from A, party B finds a pair with a leftmost element consisting of an "all Quadratic residues" quintuple. B sets σ'_j to be the rightmost element of that pair.

5) B's local output is $\sigma' = (\sigma'_1, \sigma'_2, \ldots, \sigma'_5)$.

4.2. Correctness of the Protocol

We first show, that the above protocol is correct; namely that the output satisfies the specification conditions.

Claim 4: The permutation σ' locally output by B equals $\tau \cdot \sigma \cdot \tau'$.

Proof: Clearly, in step 2 A correctly computes $E(\tau \cdot \sigma)$ We now show that σ'_j computed by B in step $4.j$ equals i_j, for every j $(1 \leq j \leq 5)$. The coordinate-wise multiplication of e_j and $E(l)$ equals (Q, Q, Q, Q, Q) if and only if the j-th element of (the quintuple representing the permutation) $\tau\sigma$ equals l. Thus, σ'_j is set to τ'_l, where l is the j-th element of $\tau\sigma$. It follows that σ' equals $(\tau \cdot \sigma) \cdot \tau'$. □

4.3. Maximum Privacy of the Protocol

We now prove that the above protocol has the maximum privacy property. First we use the Intractability Assumption of subsection 2.6 to prove that B preserves privacy with respect to any A^*, and next we prove that A preserves privacy with respect to any B^* The proofs use ideas similar to those used in the proofs of Claims 2 and 3.

Claim 5: Assuming intractability of Quadratic Residuosity (as in subsection 2.4), the above protocol preserves privacy with respect to A.

Proof's Sketch: To prove the claim, we construct a machine M that on input τ, τ', m, y and a restricted polynomial-time computable auxiliary input z, (see (2) at subsection 3.4) outputs the inputs together with the encryption of the identity permutation. As in the proof of Claim 2, ability to distinguish this output from the contents of A's (input and read-only communication) tapes will be converted to a contradiction of the Intractability Assumption of Quadratic Residuosity. □

We now prove preservation of privacy with respect to B. This time we use no intractability assumptions.

Claim 6: The protocol preserves privacy with respect to B.

Motivation to the Proof: What does B learn from the $\bar{E}(i_j)$'s? Since A uses independently chosen probabilistic encryptions in each step $3.j$, we concentrate on what is learned from $\bar{E}(i_1)$. B has received five pairs, the left element of one of them is (Q,Q,Q,Q,Q) while the left elements of the other pairs are quintuples with three Q's and two N's. It is crucial that the location of the N's in these quintuples is "random" and thus does not leak any information.

Proof Sketch: We will show that there exist a machine M which on input σ, m, y, z and σ' $(=\tau\sigma\tau')$, outputs a probability distribution which is identical to the distribution on B's input and read-only tapes during interaction with A. M operates as follows

for j=1 to 5 do begin

1.j) M constructs five pairs $(v_{j,1},1),(v_{j,2},2),\dots,(v_{j,5},5)$, where the quintuple $v_{\sigma'_j}$ is (a random) (Q,Q,Q,Q,Q) and all the other quintuples have three (random) quadratic residues in (independent) randomly selected locations and (random) quadratic non-residues in the remaining locations.

2) M outputs his inputs and all (25) pairs constructed in step 1.

One can easily show that the probability distribution formed by M is identical to the distribution of B when interacting with A. The claim follows. □

4.4. Summing Up

Combining Claims 4, 5 and 6, we get

Theorem 2: The above protocol is a maximum privacy protocol for the permutation switching problem.

ACKNOWLEDGEMENTS

We are grateful to Ilan Newman for listening and commenting on the work, while in progress.

REFERENCES

[Bar] Barrington, D.A., "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1", *Proc. 18th STOC*, 1986, pp. 1-5.

[CGMA]Chor, B., S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *Proc. 26th FOCS*, 1985, pp. 383-395.

[Coh] Cohen, J.D., "Secret Sharing Homomorphisms: Keeping Shares of a Secret", technical report YALEU/DCS/TR-453, Yale University, Dept. of Computer Science, Feb. 1986. Presented in *Crypto86*, 1986.

[DH] Diffie, W., and M.E. Hellman, "New Directions in Cryptography", *IEEE Trans. on Inform. Theory*, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[EGL] Even, S., O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts", *CACM*, Vol. 28, No. 6, 1985, pp. 637-647.

[GMW1]Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS*, 1986.

[GMW2]Goldreich, O., S. Micali, and A. Wigderson, "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority", *Proc. 19th STOC*, 1987.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs", *Proc. 17th STOC*, 1985, pp. 291-304.

[HM] Haber, S., and S. Micali, private communication, 1986.

[Y1] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.

[Y2] Yao, A.C., "How to Generate and Exchange Secrets", *Proc. 27th FOCS*, 1986.

MULTIPARTY COMPUTATIONS ENSURING PRIVACY OF EACH PARTY'S INPUT AND CORRECTNESS OF THE RESULT

David Chaum
Ivan B. Damgård
Jeroen van de Graaf

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam the Netherlands

Summary

A protocol is presented that allows a set of parties to collectively perform any agreed computation, where every party is able to choose secret inputs and verify that the resulting output is correct, and where all secret inputs are optimally protected.

The protocol has the following properties:

- One participant is allowed to hide his secrets *unconditionally*, i.e. the protocol releases no Shannon information about these secrets. This means that a participant with bounded resources can perform computations securely with a participant who may have unlimited computing power. To the best of our knowledge, our protocol is the first of its kind to provide this possibility.
- The cost of our protocol is linear in the number of gates in a circuit performing the computation, and in the number of participants. We believe it is conceptually simpler and more efficient than other protocols solving related problems ([Y1], [GoMiWi] and [GaHaYu]). It therefore leads to practical solutions of problems involving small circuits.
- The protocol is *openly verifiable*, i.e. any number of people can later come in and rechallenge any participant to verify that no cheating has occurred.
- The protocol is optimally secure against conspiracies: even if $n-1$ out of the n participants collude, they will not find out more about the remaining participants' secrets than what they could already infer from their own input and the public output.

- Each participant has a chance of undetected cheating that is only exponentially small in the amount of time and space needed for the protocol.
- The protocol adapts easily, and with negligible extra cost, to various additional requirements, e.g. making part of the output private to some participant, ensuring that the participants learn the output simultaneously, etc.
- Participants can prove relations between data used in different instances of the protocol, even if those instances involve different groups of participants. For example, it can be proved that the output of one computation was used as input to another, without revealing more about this data.
- The protocol can be usen as an essential tool in proving that all languages in IP have zero knowledge proof systems, i.e. any statement which can be proved interactively can also be proved in zero knowledge.

The rest of this paper is organised as follows: First we survey some related results. Then Section 2 gives an intuitive introduction to the protocol. In Section 3, we present one of the main tools used in this paper: bit commitment schemes. Sections 4 and 5 contain the notation, terminology, etc. used in the paper. In Section 6, the protocol is presented, along with proofs of its security and correctness. In Section 7, we show how to adapt the protocol to various extra requirements and discuss some generalisations and optimisations. Finally, Section 8 contains some remarks on how to construct zero knowledge proof systems for any language in IP.

1. Related Work

The problem of multiparty secure computations is an important one. When stated in its most general form, its solution implies a solution—at least a theoretical oneo almost any protocol construction problem. The first to consider this problem was Yao[Y2]. He devised protocols, that under various cryptographic assumptions could keep the inputs secret while allowing each participant only a negligible chance of cheating. He did not, however, address the problem of "fairness", i.e. ensuring that the participants learn the output simultaneously. Later, Yao[Y1] solved also this problem for the two-party case.
Recently, in work done independently from (but earlier than) ours, Goldreich, Micali and Wigderson[GoMiWi] used Yao's two-party construction to devise a multiparty solution, based on the existence of a trapdoor one-way function. This protocol implements a multiparty simulation of the computation in a circuit. Each participant holds a share of each bit in the real computation, and these shares are manipulated by using Yao's two party construction $O(n^2)$ times, where n is the number of participants. In other work done independently from and almost simultaneously with ours, Galil, Haber and Yung generalise all the properties of Yao's construction to the multiparty case, and simplify the use of

Yao's protocol in the multiparty simulation. Also, they give a concrete construction based on Diffie-Hellman key exchange rather than the existence of a trapdoor one way function. Very recently, Goldreich and Vainish [GoVa] found another simplification by designing a special purpose two-party protocol which could replace Yao's protocol in the multiparty construction.

In comparison, our protocol solves the multiparty case using a totally different concept. It attacks the problem "directly", without using Yao's two-party protocol, and without using reductions to 3-colorability [GoMiWi2] or to SAT [BrCr][Ch] to prove the validity of messages. We rely on a stronger cryptographic assumption than [GoMiWi], namely the *Quadratic Residuosity Assumption* [GoMi]. This, however, allows a simpler and more efficient construction, capable of implementing all the "primitives for cryptographic computations", defined in [GaHaYu]. Although our protocol could also be based only on the existence of a trapdoor one-way function, this would reduce the efficiency considerably. Some of the techniques upon which our protocol is based were previously developed and used in the work of Chaum[Ch] and Brassard and Crepeau[BrCr]. By specializing to the case where only one participant supplies the inputs, our protocol can provide slightly more efficient solutions to the problems solved there.

An interesting consequence of our work is that any statement which can be proved interactively, can also be proved in zero knowledge, provided that our cryptographic assumption holds (one assumption which will do in this case is the existence of clawfree trapdoor functions). More formally: any laguage in IP has a zero knowledge proof system. This result was already proved for protocols with a constant number of rounds in [GoMiWi2], where a proof of the result in full generality is attributed to Ben-Or. As far as we know, this last proof is still unpublished, so we cannot comment on possible differences between this proof and ours.

2. Overview of the Construction

For the intuitive sketch given in this section, we will make some simplifications. We will consider only two parties, A and B and we will do a computation on one AND-gate only. At the end of the section we will see how this method can be generalised to more parties, and to a computation consisting of many gates.

Note that this AND-gate computation, where both parties want to hide their input from eachother, has a meaningful application: consider the situation where Alice and Bob have just met, and each considers dating the other. Niether wishes to loose face in the following sense: if Alice wants a date but Bob doesn't, Alice does not want to let Bob know that she wanted the date. And the same holds for Bob. In other words: if a party does not want the date it does not find out the other party's decision.

Note that sometimes a party can derive the other's input just by combining his own input and the output of the joint computation. This can never be avoided, however.

The AND-gate is represented by a truth-table T. We will show what transformations A and B have to apply to T in order to compute the logical AND, such that

- both parties can be sure (with very'high probability) that the transformations are done correctly;
- both parties can keep their input hidden from the other party; and
- both parties can be sure the result of the computation is correct.

THE PROTOCOL

1.1 A applies a randomly chosen permutation, σ_j, to the rows of T.

1.2 A chooses two bits b_1 and b_3, and uses these to compute the XOR of the kth column and b_k. This procedure is known as *inversion of columns.*

1.3 A chooses four bits $d_1 \cdots d_4$, and computes the XOR of the lth entry in the last column and d_l. This procedure is known as *encrypting the output column.*

1.4 A encrypts $b_1, b_3, d_1 \cdots d_4$ in such a way that she cannot disavow these values later, while B cannot see what the value is without A's help. This is known as *commiting to the bits.*

We will indicate the final result of these steps with $\hat{T}$.

B will verify whether $\hat{T}$ was constructed correctly in the next steps:

2. A creates a transformed truthtable T' in exactly the same way as he created $\hat{T}$.

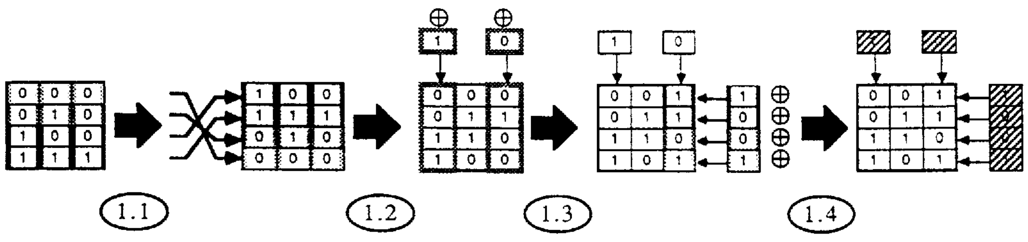


Figure 1. Party A's transformation of the AND-gate. Note that all figures show the table *after* the indicated operation has been performed.

3. *B* flips a coin.

- If the coin came out heads, *A* must reveal exactly how she constructed T', so that *B* can verify that this was done correctly.

- If the coin came out tails, *A* must show a special relation between $\hat{T}$ and T'. Details on this can be found in Section 6.1. Here, it suffices to know that if the relation holds, it implies that $\hat{T}$ was correctly constructed if and only if T' was.

After *m* repetitions of steps 2 and 3, *B* is convinced with very high probability that *A* constructed $\hat{T}$ correctly. Now *B* will do to $\hat{T}$ what *A* did to *T*, and even more:

4.1 *B* permutes the rows of $\hat{T}$, together with the commitments of the output column entries.

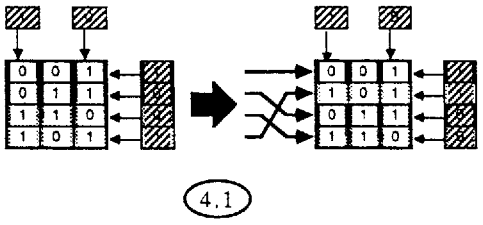


Figure 2. The row-permutation performed by *B*.

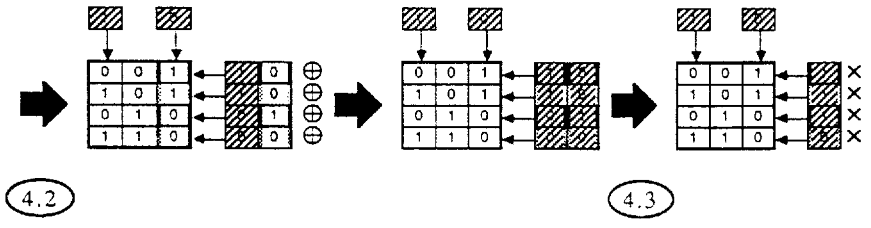


Figure 3. The blinding.

4.2 *B* flips a coin for each output column entry and applies the XOR.

4.3 *B* computes, *using A's bit commitment scheme*, a commitment containing the XOR of his own bit and the one chosen and committed to by *A* for this output column entry. Although *A* can later open the resulting commitment, he will have no idea which of his original commitments it was computed from. This requires special properties of the commitment scheme used.

This *blinding* is necessary to hide *B*'s permutation. *B* also has to do a transformation corresponding exactly to the one of *A*:

4.4 *B* chooses bits for inversion of his own input column, and for the output column.

4.5 *B* chooses bits for inversion of each output column entry.

4.6 *B* commits to the bits chosen in steps 4.4 and 4.5. He then sends his final result, T^*, to *A*.

5. In the same way as in steps 2 and 3, *B* convinces *A* that T^* was constructed correctly.

Now *A* and *B* have together constructed a double-blinded version T^* of the gate that satisfies the conditions listed before step 1. They will use T^* in performing the computation as follows:

6. Together they select the correct row of T^* by announcing their choice of input bits for T^*. These input bits are the x-or sum of the "real" input bits and the inversion bits for the corresponding columns.

7. *A* and *B* both reveal the inversion bit for the output column, and the inversion bit

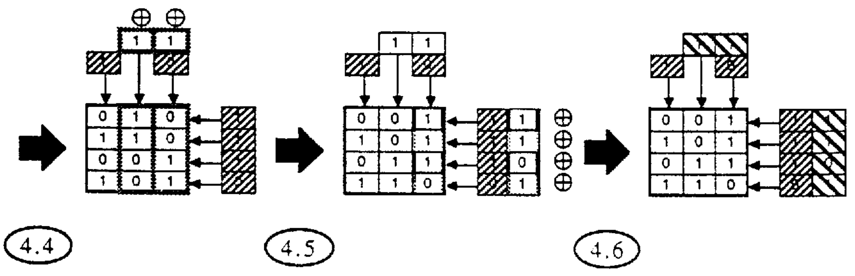


Figure 4. The final part of *B*'s transformation.

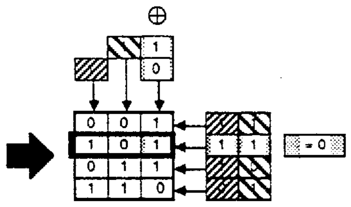


Figure 5. The computation.

for the entry in the output column which is in the selected row. By x-oring this output column entry with the revealed inversion bits, they can both compute the bit that is the result of the computation.

This basic protocol can be generalized in several ways:

- Instead of using an AND-gate, any type of gate can be used.
- Instead of two parties, any number of parties can participate in the protocol. Each new party just has to follow B's part of the protocol, and will thus have to blind the commitments of each participant who went before.
- Instead of using only one gate, a computation involving arbitrarily many gates can be done with the protocol. The only condition is that when gate T_1 is connected to T_2, then their corresponding output column and input column were inverted using the *same* bits. When later a row is selected in the encrypted version of T_1, the inversion bits that apply to the single output column entry will be revealed, while the coulumn inversion bits will remain secret. The revealed inversion bits are now x-ored with the output column entry selected, and the result can be used directly as an input bit to the encrypted version of T_2.

3. Bit Commitment Schemes

A bit commitment scheme is a tool allowing protocol participant A to "commit" to a bit $b \in \{0,1\}$ by releasing some information related to b such that:

— She cannot later convincingly claim that b had the opposite value.

— No other participant can find the value of b unless A allows this (opens the commitment).

More formally, a bit commitment scheme is a family of finite sets

$$\{\mathbf{I}_m\}_{m=1}^{\infty}$$

such that a member of $\mathbf{I}_m$ is a pair of functions

$$(f_m^{(0)}, f_m^{(1)}),$$

where

$$f_m^{(0)}: R_m^{(0)} \rightarrow S_m \text{ and } f_m^{(1)}: R_m^{(1)} \rightarrow S_m$$

where $R_m^{(i)}$ and S_m are finite sets, and the functions are polynomial (in m) time computable.

Further, there exists a probabilistic polynomial time algorithm Ω, which on input a value of the security parameter m outputs a member chosen randomly and uniformly from $\mathbf{I}_m$.

To use the commitment scheme, A will choose an instance, i.e. she will choose a value of m, run Ω and make m, $f_m^{(0)}$ and $f_m^{(1)}$ public.

To commit to a bit b, A chooses an element r uniformly in $R_m^{(b)}$ and makes public

$$BCS_{A,m}(b,r) = f_m^{(b)}(r).$$

Below, we list the properties of bit commitment schemes, that are of interest in our protocol:

Hiding the bit:

Let Δ be any probabilistic polynomial time algorithm which takes as input a commitment $BCS_{A,m}(b,r)$ and gives a one bit output $\Delta(BCS_{A,m}(b,r))$. Then for all constants c,

$$|Prob(\Delta(BCS_{A,m}(b,r))=b) - ½| < m^{-c}$$

for all sufficiently large m. This holds for all but a fraction $\epsilon(m)$ of the instances in $\mathbf{I}_m$, and as a function of m, $\epsilon(m)$ vanishes faster than any polynomial fraction.

Thus, guessing from a commitment which bit it contains is a hard problem. Note that b must be uniquely determined from $BCS_{A,m}(b,r)$ and r, but not necessarily from $BCS_{A,m}(b,r)$ alone. This also means that b may be impossible to find from a commitment to b, even with infinite computing power.

Unforgeability:

Let Δ be a probabilistic polynomial time algorithm which takes as input a commitment $BCS_{A,m}(b,r)$, b and r, and gives as output an element r_Δ in $R_m^{(b\oplus 1)}$. Then for any constant c,

$$Prob(BCS_{A,m}(b,r) = BCS_{A,m}(b\oplus 1, r_\Delta)) < m^{-c}$$

for all sufficiently large m. Again, this holds for all but a fraction $\epsilon(m)$ of the instances in $\mathbf{I}_m$, and as a function of m, $\epsilon(m)$ vanishes faster than any polynomial fraction.

Thus, it is not possible for a polynomially bounded A to lie about the contents of her commitments. But note that an r_Δ satisfying the condition above may not exist at all, in which case it is impossible to lie, even with infinite computing power.

Opening a commitment:

Given $BCS_{A,m}(b,r)$, A can convince anyone, that for a given value of b, she knows an r which will produce the given commitment. Typically, she will just reveal r.

Comparability:

Given two commitments, $BCS_{A,m}(b,r)$ and $BCS_{A,m}(b',r')$, there exists a Minimum Knowledge Interactive Proof (MKIP) by which A can convince anyone about the value of $b\oplus b'$.

Blinding:

Given a commitment $BCS_{A,m}(b,r)$, *another* participant, B, can choose $b' \in \{0,1\}$ and compute from this a blinded commitment $BCS_{A,m}(b\oplus b',r')$, where r' is uniformly distributed in $R_m^{b\oplus b'}$. There exists a MKIP such that, given these two commitments, B can convince anyone about the value of b'.

Moreover, we require that given two commitments $BCS_{A,m}(b\oplus b',r')$ and $BCS_{A,m}(b\oplus b'',r'')$, which have been computed by B as above, there exists a MKIP by which B can convince anyone about the value of $(b\oplus b')\oplus(b\oplus b'') = b'\oplus b''$.

Finally, it must be possible for A to open commitments which have been blinded as above, just as if she had computed them herself.

From a theoretical point of view, the opening and comparability properties do not represent demanding assumptions, since the required proofs can always be produced by using standard reductions to 3COL or SAT ([GoMiWi], [Ch], [BrCr]). We will only be concerned, however, with bit commitment schemes allowing the proofs to be produced directly and efficiently. As a consequence, the main protocol will be much more efficient.

3.1 Examples

The protocol presented in this paper relies on the the existence of bit commitment schemes satisfying the properties defined above. In this subsection, we give three examples of commitment schemes, which could be used in the protocol, namely *the Quadratic Residuosity Scheme* (QRS), *the Jacobi Symbol Scheme* (JSS) and *the Discrete Log Scheme* (DLS), of which DLS has not been used in published protocols before.

In QRS, A uses the algorithm Ω to select at random two m-bit primes, p and q with $p=q=-1$ *mod* 4. She then puts $N=pq$, $R_m^{(0)} = R_m^{(1)} = \mathbf{Z}_N^*$, and $S_m=\{x \in \mathbf{Z}_N^* \mid (\frac{x}{N})=1\}$. She makes public N and uses for example the zero knowledge protocol from [PeGr] to convince everyone that N is of the right form. In particular, this implies that -1 is a quadratic nonresidue modulo N of Jacobi symbol 1. If we define

$$f_m^{(b)}(r)= (-1)^b r^2 \ mod\ N$$

it is easy to see that a commitment will contain a 1 if and only if the commitment is a quadratic nonresidue modulo N.

The "hiding" condition above now translates into the well known *quadratic residuosity assumption:* it is hard, even probabilistically, to distinguish quadratic residues from quadratic nonresidues modulo N (a formal statement can be found in [GoMi]). Moreover, even with infinite computing power, clearly A cannot lie about the contents of a commitment.

The opening and comparability properties are easily verified for QRS: A simply makes public a square root modulo N of

$$(-1)^b BCS_{A,m}(b,r) \quad \text{resp.} \quad (-1)^{b\oplus b'} BCS_{A,m}(b,r) BCS_{A,m}(b',r')^{-1}.$$

These proofs will be MKIPs because the commitments were chosen by A herself at random.

Blinding is also a simple process: having chosen b', B computes the new commitment as

$$(-1)^{b'} s^2 BCS_{A,m}(b,r) = BCS_{A,m}(b\oplus b',rs),$$

where s is randomly chosen from $\mathbf{Z}_N^*$. B can show the value of b' by showing a square root of

$$(-1)^{b'} BCS_{A,m}(b\oplus b',rs)(BCS_{A,m}(b,r))^{-1},$$

which equals s. s is called *the blinding factor.*

Also, given two blinded commitments as above, B can show a square root of

$$(-1)^{b'\oplus b''} BCS_{A,m}(b\oplus b',r')(BCS_{A,m}(b\oplus b'',r''))^{-1}$$

to convince everyone about the value of $b' \oplus b''$.

Clearly, A can open any commitment by using her knowledge of the factors of N to compute square roots. But it is important to note that it will only be safe for her to do so, if she is convinced that she will always be opening blinded forms of commitments chosen by *herself.* This will be ensured by the construction of our main protocol.

In the JSS (first introduced by Blum in [Bl]), some participant B chooses a modulus N of the same form as above, such that the factorisation is unknown to A, and convinces her as above that N was correctly chosen. We then put

$$R_m^{(0)} = \{x \in \mathbf{Z}_N^* \mid (\frac{x}{N})=1\},$$

$$R_m^{(1)} = \{x \in \mathbf{Z}_N^* \mid (\frac{x}{N})= -1\},$$

$$\text{and } S_m = QR(N),$$

where $QR(N)$ denotes the set of quadratic residues modulo N. Finally, we set

$$f_m^{(b)}(r) = r^2 \ mod \ N,$$

where $r \in R_m^{(b)}$.

This scheme hides the bits unconditionally: Since each square modulo N has two square roots of Jacobi symbol $+1$ and two of Jacobi symbol -1, release by A of $BCS_{A,m}(b,r)$ gives away no Shannon information about b. On the other hand, the unforgeability property only holds conditionally here: if A could factor N, she could lie about the value of b.

JSS also satisfies the comparability property: Given two commitments $s_1 = BCS_{A,m}(b,r)$ and $s_2 = BCS_{A,m}(b',r')$, A can make public a square root of $s_1(s_2)^{-1}$ of Jacobi symbol $(-1)^{b \oplus b'}$ to convince everybody about the value of $b \oplus b'$. Note that this is still information theoretically secure: after having seen a proof that $b \oplus b' = 1$, say, the two possibilities $(b,b') = (1,0)$ and $(b,b') = (0,1)$ remain equally likely, even to someone who knows the factorisation of N.

Note that if A is going to commit to bits using JSS in a multiparty protocol, every participant except A must supply a different modulus, and when committing to a bit, A must make public a set of commitments to this bit, one for each modulus supplied. This is to prevent the possibility that A breaks the unforgeability property by conspiring with somebody who knows the factorisations. This of course means that the other participants must be convinced that the same bit is contained in all commitments in a given set. Fortunately, the construction of our main protocol is such that this is automatically ensured, whenever JSS is used.

Note also that if absolute certainty is needed about the correctness of the choice of N, B could be required to simply reveal the factorisation of N after the protocol has been executed. It would be too late for A by then to use this knowledge in the protocol. Another possibility is for B to prove to A in zero knowledge that the modulus has the right form. However, all the known proofs of this leave an exponentially small probability that B's claim is false, in which case A *would* be giving away information about her secrets. Thus it seems that with this scheme, A will never be quite sure that her secrets have not been betrayed until after it happened!

The next example, the Discrete Log Scheme, solves this problem. Here, B chooses an m-bit prime p, a generator g of $\mathbf{Z}_p^*$, and an element $a \in \mathbf{Z}_p^*$ at random and reveals them to A. We now put $R_m^{(1)} = R_m^{(0)} = S_m = \mathbf{Z}_p^*$, and

$$f_m^{(b)}(r) = a^b g^r.$$

This scheme also hides the bits unconditionally, since the bit contained in a commitment is not uniquely determined, if the random choice made by A is unknown. Note that if p is determined such that the factorisation of $p-1$ is known, A can easily check for herself that p is a prime and that g is a generator.

To open a commitment, A must show the discrete log base g of

$$a^{-b} BCS_{A,m}(b,r),$$

namely r. It is also easy to see that the unforgeability property holds, assuming that A cannot find the discrete log of a.

Given two commitments $BCS_{A,m}(b,r)$ and $BCS_{A,m}(b',r')$, A can prove that $b \oplus b' = 0$ by showing the discrete log of

$$BCS_{A,m}(b,r)(BCS_{A,m}(b',r'))^{-1}$$

and to show that $b \oplus b' = 1$, she shows the discrete log of

$$a^{-1} BCS_{A,m}(b,r) BCS_{A,m}(b',r').$$

Once again, it is easy to see that A cannot make a false claim about $b \oplus b'$ unless she can find the discrete log of a.

An additional advantage of DLS compared to JSS is that it is not necessary in a multiparty application that every participant different from A chooses his own instance, and that A then must commit to a bit "in parallel" using all the published instances. Since there is no trapdoor present in DLS, all participants can collaborate in choosing p, g and a, using some multiparty coinflipping protocol.

4. Protocols: notation and definitions

We think of a protocol as occurring among a set of communicating probabilistic Turing machines $\{P_1, \ldots, P_n\}$ called the participants. The protocol is described as a specification of the program each participant should follow.

Each machine has private input, output and random tapes, and one communication tape. The contents of the input tape of P_i is called ip_i, the contents of the output tape after execution of the protocol is called op_i. All machines are clocked by a common clock, and the protocol proceeds in rounds. In each round, exactly one machine can do some computation and perhaps write a message on the other machines' communication tapes. By definition, all messages sent in the protocol are written simultaneously on all communication tapes. If a machine receives a message which does not obey the constraints specified in the protocol, it stops immediately, and we say that *cheating has been detected*.

We assume the existence of a mutually trusted source of random bits. Such a source could easily be implemented using e.g. Blum's coin flipping protocol [Bl]. But since the construction of such a subprotocol is independent of our main protocol, we have chosen to abstract away the fact that a subprotocol is used.

The protocol takes some global input, known to all machines:

- A specification of the computation to be performed, in the form of a boolean circuit C.
- The cryptographic security parameter m.

Note that the security and the certainty of correctness may be varied *independently* of the complexity of the computation modelled by C. Moreover, the protocol never uses any special properties of C, such as membership of a polynomial size family. For these reasons, we look at C just as a single fixed circuit.

Of course, this does not mean that the behaviour of e.g. the running time of the protocol as a function of the size of C is not interesting, only that such dependencies should be considered independently of the security level.

4.1 Correctness and Security

The definitions in this section are straightforward generalisations of the definition of "zero knowledge proof systems" found in [GoMiRa]. This becomes clear if one takes a slightly different point of view of the situation where a prover convinces a verifier about the membership of a string x in an NP-language L:

In fact the prover and the verifier are doing an interactive computation, where only the prover supplies input, namely a certificate of membership for x. The notion of correctness of the protocol they execute corresponds to the fact that we have a proof system: we want the protocol to produce the correct output of the computation, namely "accept" if indeed the prover put in a valid certificate of membership for x. The notion of security corresponds to the fact that the proof system is zero knowledge: we want the verifier to gain nothing useful about the secrets of the prover, other than the fact that x is in L.

If we model the computation needed to verify membership for x by a boolean circuit C, it is clear that this is just a special case of the general problem addressed in this paper.

We allow C to be a probabilistic circuit, i.e. take as input a number of bits chosen uniformly from $\{0,1\}$. Thus, given the inputs $I=(ip_1, \ldots, ip_n)$, C determines a probability distribution OP_I^C on the output. We assume for simplicity that the output is public, i.e. the same output string will appear on the output tape of any participant following the protocol (the generalisation to different (private) outputs is easy and will be discussed later). Given a choice of inputs as above, we let $OP_{I,m}^{prot}$ denote the probability distribution according to which the output string of a participant following the protocol is distributed, when the protocol is executed with security parameter value m and no cheating is detected. We would like this distribution to be essentially equal to OP_I^C. In other words, the only way for (a set of) dishonest participant(s) to prevent the correct result from being computed is to stop the protocol e.g. by deliberately sending an incorrect message.

We let $OP_I^C(op)$ (resp. $OP_{I,m}^{prot}(op)$) denote the probability that the binary string op is produced as output.

Definition 4.1.

A protocol is said to be *correct* if for any choice of inputs $I=(ip_1, \ldots, ip_n)$, and any binary string op and constant c,

$$| OP_I^C(op) - OP_{I,m}^{prot}(op) | < m^{-c}$$

for all sufficiently large m□

A *conspiracy* is a subset X of $\{P_1, \ldots, P_n\}$ with $|X| < n$. The machines in a conspiracy may follow any (polynomial time) program, and they may establish private communication channels to share all the information available to them.

Intuitively we will consider the protocol secure if no conspiracy will learn more from the protocol, than what could already have been inferred from the output of the computation and the part of the input known to the conspiracy.

To state this more formally, we need some definitions:

The binary string, which is the concatenation of all messages sent during an execution of the protocol, is called a *protocol conversation.*

Let ra be a binary string containing as many bits as there are random input bits to C. Then by $f_C(ip_1, \ldots, ip_n, ra)$ we denote the output of C resulting from inputs $(ip_1, \ldots, ip_n)$ and random choice ra.

We let $P_{prot}^{(m)}(ip_1, \ldots, ip_n, ra)$ denote the probability distribution according to which a protocol conversation with security parameter m and inputs $(ip_1, \ldots, ip_n, ra)$ is distributed.

Note that this distribution is conditional on the programs followed by the machines in X, i.e. the conspiracy's strategy. Note also that the definition talks about the inputs of machines in X. This is not necessarily a meaningful concept: nothing prevents a dishonest participant from pretending that the contents of its input tape is different from what in fact it is. This "problem" is easily solved, however, if we jump ahead in the protocol description a little: Each participant must publish bit commitments, which determine a particular choice of input. We therefore *define* the input of a participant to be the bit-string committed to during the execution of the protocol.

A *simulator for* X is a polynomial time Turing machine M_X, which on input $(m,\ f_C(ip_1, \ldots, ip_n, ra),\ \{ip_i \mid P_i \in X\})$ will generate a protocol conversation distributed according to the distribution $P_{M_X}^{(m)}(f_C(ip_1, \ldots, ip_n, ra),\ \{ip_i \mid P_i \in X\})$.

Definition 4.2.

The protocol is said to be *secure against the conspiracy* X, if there exists a simulator for X, M_X, such that for any $(ip_1, \ldots, ip_n)$ and ra, the two ensembles of probability distributions

$$\{P_{M_X}^{(m)}(f_C(ip_1, \ldots, ip_n, ra),\ \{ip_i \mid P_i \in X\})\}_{m=1}^{\infty}$$

and

$$\{P_{prot}^{(m)}(ip_1, \ldots, ip_n, ra)\}_{m=1}^{\infty}$$

are polynomially (in m) indistinguisable□

We assume that the reader is familiar with this notion. For details, refer to [GoMiRa].

The protocol is said to be *secure,* if it is secure against any conspiracy.

This definition means that no matter which type of hostile behaviour is exhibited by X, the private inputs of all honest participants *and* the random inputs are optimally protected. But note that proving the protocol secure under this definition does not exclude the possibility that a conspiracy, by stopping the protocol in some clever way, could find out about the output, while preventing honest participants from getting it. The basic protocol as described in section 5 already offers protection against this, but a lot of variations on the theme are possible. More details will be given in Section 6.

5. Circuits, terminology and notation

We think of a boolean circuit C in the usual way as an acyclic directed graph, the nodes are called *gates,* the edges are called *wires.*

A wire may be connected only into a gate, in which case it is called an *input wire.* If it leaves a gate without connecting into another, we call it an *output wire.*

The input wires are partitioned into $n+1$ disjoint subsets, $I_1, \ldots, I_n, R$. I_j corresponds to the secret input chosen by P_j, such that ip_j contains one bit for each wire in I_j. In a similar way, R corresponds to the random inputs to C.

For each gate, a function is specified that maps the input bits to a one bit output. For the gate G, this function is given by *the truth table* $T(G)$. $T(G)$ is a $0-1$ matrix with $t+1$ columns and 2^t rows, where t is the number of wires connecting into G. The rows, apart from the last entry, contain each possible assignment of bits to the input of G exactly once, and the last column contains the corresponding output bits. Note that each input column in each truth table corresponds to exactly one wire in C, and that an output column may correspond to any number of wires ("fan-out" from a gate is allowed). Thus, two columns may correspond to the same wire. Two such columns are said to be *connected*.

Definition 5.1.

A *computation* Γ in C consists of a *selection* of exactly one row in each truth table. The *input of* Γ is the corresponding assignment of bits to the input wires of C, the *output of* Γ is the corresponding assignment of bits to the output wires.

A computation is said to be *consistent,* if the following is satisfied: whenever two truthtable-columns c_1 and c_2 are connected, the entry selected by Γ in c_1 equals the entry selected in c_2 □

2 Transformations of Truthtables

Our protocol works with "transforms" of truthtables. Basically, a transform of a truthtable $T(G)$ is just a row-permuted version of $T(G)$ with the last column changed and some extra (encrypted) information about the transformation included.

Definition 5.2.

Let G be a gate in C with t input wires. For $i = 1 \cdots n$, we define an *i-transform* T *of* $T(G)$ to be a $(t+1)$ by 2^t, $0-1$ matrix with a list of bit commitments attached to each row and each column. The list attached to row l is called *rlist*(l,T). The list attached to column k is called *clist*(k,T).

Each list contains one commitment from each participant P_j, for which $j \leqslant i$, except *clist*(k,T), when the k'th column corresponds to an input wire in some I_j. These lists are empty if $j > i$, and contain one bit commitment from P_j if $j \leqslant i$.

By *rsum*(l,T) (resp. *csum*(k,T)) we denote the x-or sum of all bits committed to in *rlist*(l,T) (resp. *clist*(k,T)). The sum of an empty list is defined to be 0.

A 0-transform of $T(G)$ is defined to be simply $T(G)$ itself□

Most of the work done in the protocol by P_i will consist of receiving $(i-1)$-

transforms and from these creating i-transforms. We require that there is some specific relation between a truthtable $T(G)$ and its transforms used in the protocol. When this relation holds, the transform is said to be *valid*. A valid transform can be seen as an encrypted version of $T(G)$: the rows are permuted, and each column is x-ored with an independently chosen bit. In addition to this, the last entry in each row is x-ored with an independently chosen bit. The attached lists of bit commitments (when opened) contain complete information about how the transform was constructed. More specifically we have the following:

Definition 5.3.

A *valid i-transform T of* $T(G)$ is an i-transform of $T(G)$ which satisfies that:

There exists a permutation σ of $\{1, \ldots, 2^t\}$ such that:

$$\begin{aligned} T_{lk} &= T(G)_{\sigma(l)k} \oplus csum(k,T), && \text{if } k<t+1, \\ T_{lk} &= T(G)_{\sigma(l)k} \oplus csum(k,T) \oplus rsum(l,T), && \text{if } k=t+1, \end{aligned}$$

where T_{lk} (resp. $T(G)_{lk}$) is the entry in the l'th row and k'th column of T (resp. $T(G)$)□

6. The Protocol

The first step in the protocol is that each participant chooses an instance of a bit commitment scheme and uses the protocols mentioned in Section 2 to convince all other participants that the choice was made correctly. We assume for simplicity that all participants use the same value of m as security parameter, but in principle, different participants could choose different levels of security.

The commitment scheme chosen by participant P_i is called BCS_i.

All commitment schemes BCS_i with $i<n$ must satisfy all properties defined in Section 2, so they can be instances of QRS, but not of JSS or DLS. c_n must satisfy all the properties from Section 2, except the blinding property, so it could be an instance of JSS or DLS.

The main part of the protocol proceeds in two phases: 1) The Encryption Phase and 2) The Computation Phase. Each is described formally in separate subsections below.

6.1 The Encryption Phase

The procedure TRANSFORM CIRCUIT below is iterated once by each participant, such that the i'th iteration is executed by P_i. After each iteration the protocol CHECK TRANSFORMATION is executed to verify that the preceding iteration of TRANSFORM CIRCUIT has been performed correctly. The input to the first iteration will be the truthtables in the original circuit.

PROCEDURE *TRANSFORM CIRCUIT.*

Input: in the i'th iteration, one $(i-1)$-transform of each truthtable in C. Output: one i- · transform of each truthtable in C.

For each $(i-1)$-transform S supplied in the input, do the following:

1) Suppose the corresponding gate in C has t input wires, so that S has 2^t rows and $t+1$ columns. Choose a permutation σ of $\{1, \ldots, 2^t\}$ at random and apply σ to the rows of S with attached lists. Call the result T.

2) For $l=1 \cdots 2^t$, and each $j<i$, do:

   choose a bit $s_{ij}(l,T) \in \{0,1\}$ at random and x-or the last entry in the l'th row of T with $s_{ij}(l,T)$. Find the commitment

   $$BCS_j(b_j(l,T), r_j(l,T)) \in rlist(l,T),$$

   and use the blinding property of BCS_j to replace this commitment with

   $$BCS_j(b_j(l,T) \oplus s_{ij}(l,T),\ r'_j(l,T)) \in rlist(l,T).$$

3) For $k=1 \cdots t+1$ do:

   if the kth column in T does not correspond to an input wire in I_i, then do nothing.

   otherwise, choose a bit $b_i(k,T) \in \{0,1\}$ in the following way: If column k in T is connected to another column in another transform for which a bit b has already been chosen, then put $b_i(k,T)=b$. Otherwise, choose $b_i(k,T)$ at random. Now x-or all entries in the k'th column of T with $b_i(k,T)$, and append to $clist(k,T)$ a commitment

   $$BCS_i(b_i(k,T), r_i(k,T)).$$

4) For $l=1 \cdots 2^t$ do:

   choose a bit $b_i(l,T) \in \{0,1\}$ at random, x-or the last entry in the l'th row of T with $b_i(l,T)$, and append to $rlist(l,T)$ a commitment

   $$BCS_i(b_i(l,T), r_i(l,T)).$$

Notice that the special way of choosing $b_i(k,T)$ in step 3 ensures that whenever two columns are connected, the contents of the corresponding column lists will be identical.

The purpose of step 2 is to hide the row permutation σ. The blinding in this step ensures that the contents of the row lists in S and T are statistically independent, so that no information about σ is revealed by the row lists. The reader may have noticed that information about σ *is* revealed by the contents of the input columns in T. But since each such column contains as many 0's as 1's, a particular entry in a column still has the same probability of being the encryption of a 1 as of a 0, if the bit x-ored into the column by P_i is unknown. This will be sufficient to make the protocol secure, as we shall see.

The interactive proof that every T was created according to the protocol proceeds as follows:

PROTOCOL *CHECK TRANSFORMATION*

Input: For every truthtable in C, an $(i-1)$-transform S and an i-transform T, which P_i claims was created correctly from S.

1) For every corresponding pair (S,T) in the input, P_i creates from S another i-transform T' in exactly the same way as T, but with new independent choices of permutation and bits. T' is made public.

2) The mutually trusted source of random bits chooses $b \in \{0,1\}$ at random.

3) If $b=1$, P_i must for every T': make public the row permutation used for creating T', open all his bit commitments attached to T', and reveal all blinding factors (see Section 2) he used in blinding other commitments. This allows everybody to check that T' was correctly constructed.

4) If $b=0$, P_i must show for all corresponding (T,T') a relation between T and T':

   P_i makes public the permutation $\pi=\sigma(\sigma')^{-1}$, where σ (resp. σ') is the permutation used in creating T (resp. T').

   For $k=1 \cdots t+1$: if P_i appended commitments to *clist*(k,T) and *clist*(k,T'), P_i uses comparability of BCS_i to show from his commitments the value of

$$b_i(k,T) \oplus b_i(k,T').$$

   For $l=1 \cdots 2^t$, P_i uses comparability of his commitments in *rlist*(l,T) and *rlist*$(\pi(l),T')$ to show the value of

$$b_i(l,T) \oplus b_i(\pi(l),T').$$

   For $l=1 \cdots 2^t$, and each $j<i$, P_i uses the blinding property of BCS_j to show the value of

$$s_{ij}(l,T) \oplus s_{ij}(\pi(l),T').$$

   This allows everybody to check that the following holds:

$$\begin{aligned} T_{lk} &= T'_{\pi(l)k} \oplus csum(k,T) \oplus csum(k,T'), && \text{if } k<t+1, \\ T_{lk} &= T'_{\pi(l)k} \oplus csum(k,T) \oplus csum(k,T') \oplus rsum(l,T) \oplus rsum(\pi(l),T'), && \text{if } k=t+1. \end{aligned}$$

Steps 1 - 4 are repeated m times.

Theorem 6.1.

For every $i = 1 \cdots n$, the probability that P_i has cheated (i.e. sent incorrect messages) anywhere in the protocol without this being detected is smaller than m^{-c} for any constant c and all sufficiently large m. If no cheating occurs, then the encryption phase ends with P_n outputting a valid n-transform of every truthtable in C.

Proof.

It is not hard to see that if for at least one iteration in the CHECK TRANSFORMATION protocol, P_i was able to give satisfying answers in *both* steps 3 and 4, and if P_i was not able to change the contents of his bit commitments, then T must have been correctly constructed. This and the unforgeability property of all bit commitment schemes used suffices to prove the first statement. The second follows from the obvious fact that if T was created correctly from S in the TRANSFORM CIRCUIT protocol, then T is valid if S was valid$\square$

If all participants use QRS for bit commitments, then the unforgeability property holds unconditionally for all participants, which means that the probability of undetected cheating will be at most 2^{-m}.

Intuitively, the CHECK TRANSFORMATION protocol is secure from P_i's point of view because all the other participants never see anything but random "copies" of S or T, which they could also have produced themselves. All other participants will therefore be convinced that T was correctly constructed, but will learn nothing more about T. A formal proof can be found in Section 6.3.

6.2 The Computation Phase

At the end of the encryption phase, P_n has output a set of n-transforms of the truthtables in C. This set will be called C^*.

Let w be an input wire in C connecting into the gate G, and suppose w corresponds to column k in $T(G)$. Let T^* be the n-transform of $T(G)$ contained in C^*.

As the first step in the computation phase, *encrypted* input bits are specified for each such w. This is done as follows:

if $w \in R$, then the mutually trusted source of random bits selects a bit b_w, which is made public.

if $w \in I_i$ for some i, P_i reads a bit b in ip_i corresponding to w from its private input tape, and then makes $b_w = b \oplus b_i(k, T^*)$ public.

To describe the next steps in the computation phase, we need to define the notion of computations in C^*.

Definition 6.1.

By a *computation* Γ^* *in* C^*, we mean a selection of exactly one row in each n-transform in C^*.

A computation Γ^* is said to be *consistent,* if the following is satisfied:

Suppose output column k_1 in the transform T_1^* is connected with column k_2 in the transform T_2^*, and let $\Gamma^*(k_1)$ and $\Gamma^*(k_2)$ be the entries selected by Γ^* in the two columns. Then

$$\Gamma^*(k_1) \oplus rsum(l_1, T_1) \;=\; \Gamma^*(k_2),$$

where l_1 is the index of the row selected by Γ^* in T_1^* $\Box$

The intuition behind these rather technical looking definitions is very simple: if we assume that all transforms in C^* are valid, then the selection of a row in a table T^* corresponds to selecting a row in the original truthtable, under the product of all row permutations chosen for T^* by the participants. Therefore, a computation in C^* corresponds to a computation in C. Moreover, the consistency condition on a computation in C^* just says that if we look at the string of bits selected in the computation and "remove" all layers of encryption, then the consistency condition as defined in Section 5 holds for the computation in C (note that by construction of the n-transforms in C^*, $csum(k_1, T_1^*) = csum(k_2, T_2^*)$). Formally, we have the following:

Given an n-transform T^* in C^*, let σ_i be the row permutation chosen by participant P_i for T^*. If Γ^* is a computation in C^* selecting row l in T^*, then we define *the corresponding computation* Γ in C by selecting row $(\sigma_n \cdots \sigma_1)^{-1}(l)$ in the original truthtable.

Lemma 6.2

Let Γ^* be a computation in C^* and suppose all transforms in C^* are valid. Then the corresponding computation Γ in C is consistent if and only if Γ^* is consistent.

The rest of the protocol is now just an algorithm which constructs a consistent computation in C^* from the input as specified above.

PROCEDURE *CONSTRUCT COMPUTATION*

Input: The fully encrypted circuit C^* *and for each input wire* w *an attached input bit* b_w *specified as above. Output: A consistent computation* Γ^* *in* C^* *and its output.*

1) Mark every input wire.

2) For every gate G for which all input wires are marked, do the following:

   Let T^* be the n-transform in C^* of $T(G)$. Let l be the index of the row in T^*, whose first entries match the bits attached to the input wires of G, and let b_l be the last entry in this row.

Record row number l as selected by Γ^*. Open all bit commitments in $rlist(l,T^*)$, and put

$$b_l^* = b_l \oplus rsum(l,T^*).$$

3) For every wire w connecting out of G, mark w and attach to it

$$b_w = b_l^*.$$

4) If any wires in C are still unmarked, go to Step 2).

5) For every output wire w, do the following:

suppose w corresponds to output column k in T^*. Then open all commitments in $clist(k,T^*)$, and compute the final result for this wire as:

$$Result(w) = b_w \oplus csum(k,T^*).$$

The reader can easily verify that if all participants supply the information needed in step 2, then the procedure runs in time linear in the number of gates in C, and constructs a consistent computation Γ^*. Furthermore, it is easy to see that step 5 above in fact produces the output of the computation in C corresponding to Γ^*.

6.3 Proofs of Correctness and Security

Before starting on this section, the reader is well advised to review the definitions of correctness and security given in Section 4.1.

Using the results proved in preceding sections, it is now not hard to prove

Theorem 6.3

The protocol is correct, as defined in definition 4.1.

Proof

First note that for each input wire w in R, the input bit given to C is the x-or sum of one bit chosen by the mutually trusted random source, and one bit chosen by each participant (namely the bit committed to in the *clist* corresponding to w). By Lemma 6.2, this means that the conditional probability distribution of the output of the protocol assuming that no cheating has occurred, is exactly equal to the "right" probability distribution OP_C^I, for any choice I of inputs. The theorem now follows from Theorem 6.1 and elementary probability theory□

To show that the protocol is secure, we assume the existence of some conspiracy X.

We must then exhibit a simulator M_X for X and prove that its output is polynomially indistinguisable from an actual protocol conversation. Only informal descriptions will be given, we trust that this will make the proofs easier to understand, and that the reader will have no trouble in filling in the necessary details.

We begin with the description of the simulator M_X:

Recall that M_X is given as input the security parameter value m, the output of a "real" computation $f_C(ip_1, \dots, ip_n, ra)$, and the part of the input "known" to X, $\{ip_i \mid P_i \in X\}$. In addition to this, M_X is of course allowed to use the machines in X in any (feasible) way it likes. We will describe the algorithm of M_X as a simulated protocol execution, where the participants in X act "as themselves" and M_X plays the parts of all other participants.

M_X starts by putting ip_i on the input tape of P_i for every $P_i \in X$. Also, the random tapes of all machines in X are filled in using M_X's own random tape. This means that all the machines in X are "deterministic" from now on.

For simplicity, we assume that all participants use QRS for bit commitments. M_X therefore chooses a modulus N_i for all participants P_i not in the conspiracy, while the participants in X choose their own moduli. Although this means that M_X in fact knows some of the factorisations involved, it is of course essential that this is not used in the simulation!

Next, M_X executes a simulation of the proof that N_i is a Blum integer, for every $i = 1 \cdots n$. When i is such that $P_i \in X$, P_i can be used directly as the prover, otherwise the simulation from [GrPe] can be used.

We now come to the executions of the TRANSFORM CIRCUIT procedure. If $P_i \in X$, then the i'th iteration of this procedure is replaced by an execution of the procedure SIMULATE CONSPIRACY below, otherwise the procedure SIMULATE HONEST PARTICIPANT is used.

PROCEDURE *SIMULATE HONEST PARTICIPANT.*

Input: one (i-1)-transform of every truthtable in C. Output: one (i-1)-transform of every truthtable in C.

i) For every $(i-1)$-transform S supplied in the input, which does not correspond to an output gate, M_X creates an i-transform T exactly according to the protocol.

ii) For every $(i-1)$-transform S corresponding to an output gate, an i-transform T is created in exactly the same way as in step i. In addition to this, the following is done:

    Recall that for each output gate G of C, a bit b_G is specified in the input to M_X, as part of the given value of $f_C(ip_1, \dots, ip_n, ra)$. The simulated computation must produce this bit as output from G. This is ensured by simply multiplying the commitment from P_i in some of the row lists by -1, so that the last column of T

becomes an encryption of a column with all entries equal to b_G.

Note that the above modification only has to be done once, even if the simulation is done with more than one honest participant.

iii) To simulate the CHECK TRANSFORMATION protocol, M_X chooses $b_M \in \{0,1\}$ at random. If $b_M = 1$, a set of T''s is created as valid transforms from the S's, otherwise, the T''s are created from the T's as randomly chosen i-transforms such that step 4 in the CHECK TRANSFORMATION protocol can be executed.

iv) The trusted source outputs a bit b. If $b = b_M$, M_X just executes step 3 in the CHECK TRANSFORMATION protocol if $b = 1$, and step 4 if $b = 0$. This is possible by construction of T'. Otherwise, M_X rewinds all machines in X to their configuration just after the last execution of step ii. We then go back to step iii and try again with a new independent choice of b_M and T'.

Steps iii and iv are repeated until step iv has been successful m times.

Note that the expected number of "trial T''s", we must create in the above procedure before b happens to be equal to b_M is constant, and that the above procedure therefore takes expected linear time in m.

PROCEDURE *SIMULATE CONSPIRACY.*

Input: one (i-1)-transform of every truthtable in C. Output: one i-transform of every truthtable in C.

i) M_X uses P_i to compute an i-transform T of every (i-1)-transform S in the input.

ii) The first round of the CHECK TRANSFORMATION protocol is executed as follows: step 1 is executed only once, but steps 2-4 are executed several times, rewinding P_i after each iteration. This goes on until P_i has shown *both* that T is equivalent to T', and that T' was correctly constructed. If either proof is not valid, the simulator stops.

The reader can easily verify that having seen both proofs, M_X can find out exactly how T was constructed from S. In particular, all the blinding factors used by P_i can be found. This means that M_X can now open all bit commitments used sofar in the simulation, without making further use of the machines in the conspiracy.

iii) The rest of the rounds in the CHECK TRANSFORMATION protocol are executed exactly as in an actual protocol execution.

Once again, it is easy to see that the above procedure takes expected polynomial time.

In the computation phase, M_X lets all the machines in X specify their encrypted choice of input bits, and specifies randomly chosen bits for all other participants. It is now easy to see that the CONSTRUCT COMPUTATION procedure can be executed exactly as in the protocol: By the remarks in the SIMULATE CONSPIRACY procedure, M_X has all the information it needs to open all bit commitments from participants not in X.

Why does this simulation look just like a "real" protocol conversation? Consider the set of messages sent by participant P_i in the protocol, and suppose P_i is honest. It is easy to see that the bits shown "in clear" in these messages will be distributed in exactly the same way in the simulation as in an actual protocol conversation. The difference between simulation and protocol therefore lies only in the distribution of the bits hidden in P_i's bit commitments. It is intuitively clear that to notice this difference, a distinguisher would need the ability to tell the difference between commitments containing 0's and those containing 1's. But by assumption, this cannot be done efficiently. This argument is formalised below.

Theorem 6.4.

The protocol is secure.

Proof.

It remains to be shown that the output of the simulator described above is polynomially indistinguishable from an actual protocol conversation. For simplicity, we will only do this in the case where $n-1$ participants conspire against one honest participant P_i. Cases with smaller conspiracies are in principle similar and introduce only technical differences.

Recall that a polynomial time distinguisher Δ is a probabilistic polynomial time algorithm which takes as input a binary string and gives a one bit output. Let $p_{prot}(m)$ be the probability that Δ outputs a 1 when given an actual protocol conversation with security parameter m as input. Similarly, $p_{sim}(m)$ is the probability that Δ outputs a 1 when given a simulated conversation created as above. By way of contradiction, we assume that there exists a choice of inputs $(ip_1, \ldots, ip_n, ra)$ which will result in distinguishable protocol and simulated conversations, i.e. there exists a constant c, such that

$$|p_{prot}(m) - p_{sim}(m)| > m^{-c} \qquad (1)$$

for infinitely many m.

Below, we will derive a contradiction with the quadratic residuosity assumption by showing how to use Δ to construct an algorithm that will distinguish quadratic residues from non residues modulo a Blum integer N, whenever N has m-bit prime factors, and m satisfies the above equation. Let us therefore assume that we are given an element x in $\mathbf{Z}_N^*$ of Jacobi symbol 1.

Below, we will describe a polynomial time Turing machine M_X', which on input $(ip_1, \ldots, ip_n, ra)$ and x chosen as above will generate a conversation. This conversation will be a "real" protocol conversation if x is a quadratic residue, and it will be distributed exactly as a simulated one if x is a quadratic nonresidue.

The description of M_X' is easy, based on the algorithm of M_X: M_X' assigns N as modulus to P_i, and then works exactly as M_X, except for steps i and ii in the SIMULATE HONEST PARTICIPANT procedure. In step i, M_X' will: for each commitment placed by P_i in an input column list choose a bit b at random and multiply the commitment by x^b. To see why this is done, observe that in a real protocol conversation the bits used in the (encrypted) computation and the bits contained in column lists are correlated: they allways x-or together to the bits used in the unencrypted computation. If x is a quadratic nonresidue, this correlation is destroyed by the multiplications by x, corresponding to the fact that we should be producing a simulated conversation in this case. The resulting transformation, however, is no longer valid. Since both simulation and protocol coversations produce valid transforms for non output gates, we have to do something about this. But it is not hard to see that the correctness of the transform can be ensured by also multiplying appropriately chosen commitments in row lists and output column list by x. Note that M_X' will know how to this, since it knows exactly how every transform was computed. Finally in step ii, the multiplications by -1 are replaced by multiplications by x.

In the computation phase, M_X' specifies inputs for P_i according to the protocol, i.e as if the x's had not been multiplied in. The reader may have observed that a problem could arise here: To complete this phase, M_X' must be able to open some of the commitments from P_i placed in row lists. But this is of course impossible if the commitment has been multiplied by x. Note, however, that since all inputs are known to M_X', it is known which rows will be used in the compuation in C, and therefore it can be predicted which rows will be used in the encrypted computation. This, together with the observation that the necessary ajustments above can always be done while leaving at least one row untuched, solves the problem.

Now observe, that if x is a quadratic residue, multiplying a commitment by x does not change the bit contained in the commitment, while if x is quadratic nonresidue, the bit is complemented. Using this fact, it is straightforward to check that the claim above about the output of M_X' holds.

Let $p_{prot}(m \mid N)$ be the probability that Δ outputs a 1 when given as input a protocol conversation in which P_i chooses N as modulus. $p_{sim}(m \mid N)$ is defined similarly. An easy calculation will show that (1) implies

$$|p_{prot}(m \mid N) - p_{sim}(m \mid N)| > m^{-c+1} \tag{2}$$

for infinitely many m and for more than a negligible (i.e. polynomial) fraction of the possible choices of N. Elementary probability theory now shows that if x is a randomly chosen element in $\mathbf{Z}_N^*$ of Jacobi symbol 1, then we can guess whether x is a quadratic residue with an advantage polynomially larger than ½, whenever (2) holds. But this contradicts the hiding property of QRS, i.e. the intractability assumption on the quadratic residuosity problem□

Theorem 6.5.

If P_n is honest and uses DLS for bit commitments, then the protocol releases no information in the Shannon sense about the private input of P_n and all intermediate results in the computation.

Proof.

Since DLS hides the bits unconditionally, it is easy to see that the protocol provides even a conspiracy consisting of all other participants with nothing more than the string of input bits and intermediate results, encrypted under a true one-time pad□

A similar result holds if P_n uses JSS, *assuming* that the moduli used are of the right form. Since the known interactive proofs of this would leave a small probability that this is not so, P_n would on the average be releasing an exponentially small amount of information when using JSS.

7. Generalizations

In this section, we will show how to adopt the protocol to various additional requirements. It will be shown that a very flexible functionality can obtained from a general computation protocol, as long as it protects the private inputs, computes correct results, and allows execution of *coordinated* instances, i.e. participants can prove relations between inputs and outputs used in different instances of the protocol. In particular, a protocol which implements the "input-secure computation"-primitive as defined in [GaHaYu] and allows coordinated instances can in fact implement all four "primitives for cryptographic computation" defined in [GaHaYu]. From the comparability property of the bit commitment schemes used, it is clear that our basic protocol satisfies these requirements.

7.1 Private Outputs

How can we make some of the output bits private to a participant P_i? First, note that this problem could always be solved by rewriting the circuit, such that each of the output bits in question were x-ored with a bit chosen at random by the participant.

With our protocol, however, this is not needed: we can just modify the protocol such that P_i is not required to open his bit commitments corresponding to the output wires in question. The proof of security could easily be modified to take this into account: just note that if some output bits are private to a participant not belonging to a conspiracy, then the simulation can be done without knowing these bits.

7.2 Simultaneous Release of the Output

An obvious strategy for a set of dishonest participants would be to try stopping the protocol early in such a way that this conspiracy could find out about the output while preventing honest participants from getting it. To solve this problem in general, one could just execute a set of coordinated instances, such that the input was constant over all rounds, while the circuit was rewritten such that exactly one bit of the result was computed in each instance.

Note, however, that our basic protocol already implements a bit by bit release, since the "encryptions" of the output columns are opened for one column at a time. Thus, each participant will never have more than a one bit advantage over any other participant. Also this statement could be incorporated into the formal security proof: loosely speaking, if the protocol stops without computing the complete result, then the simulation can be done without knowing all output bits.

If simultaneous release is to be combined with private outputs, however, then a bit by bit release does not make sense because the private outputs may have different lengths. We propose instead the following solution, the basic idea of which was first introduced by Yao in [Y1]: The parties agree on a probabilistic circuit, which will produce as output an instance of a bit commitment scheme based on a trapdoor one way function, e.g an instance of QRS, together with the trapdoor information. Our basic protocol can now be used to do a computation in this circuit. The parties open the specification of the instance, such that commitments can be computed, but keep the trapdoor closed. In the case of QRS, this would mean that a public modulus has been computed, but its factorisation is still unknown to everyone. Since all coinflips in the computation are secret, no participant can compute the trapdoor information by himself. The parties now go back to the original circuit and do the computation, using the bit commitment instance computed above for all commitments in output column lists. When this is done, the trapdoor information is opened, bit by bit, which is possible by the above remarks. Using Yao's terminology, this is a *fair* protocol because all participants need exactly the same information in order to get their share of the output.

7.3 Other Special Requirements

Since our basic protocol protects not only the inputs, but also all intermediate results, a variety of special properties can be obtained by rewriting the circuit. For example, the result could be distributed only to a secret subset of users, the subset being chosen at random or based on the result. Also, a secret permutation could be applied to the private output of some participant to hide the order in which he obtains his output bits. This could important, e.g in the implementation of a game such as poker.

7.4 Verifiability

Is it possible for a non participant to check from, say, a recording of all messages sent in the protocol that the result computed is a correct one? It is clear that such a check cannot be possible without interacting with the participants, since otherwise the protocol could not be minimum knowledge with respect to the private inputs: recall that there should be no way for anyone to tell whether he is presented with a genuine protocol conversation or a simulated one. But this is exactly what we would be asking the non participant above to do!

With interaction, however, checking is easy: each participant could be asked to prove once again using the CHECK TRANSFORMATION protocol that he has computed valid transforms. Then all of the computation phase could be checked without interaction.

7.5 Fault Recovery

In our basic construction, there is not much one can do short of stopping the protocol, if some participant sends incorrect messages, or just stops completely. In the recent literature, a number of techniques have been proposed for recovering from such situations.

Under the assumption that the majority of users are honest, [GoMiWi] propose to verifiably secret share the secret inputs of all participants. In the event of a fault, a majority of users would recover all the secrets of the participant who stoped, and complete the computation. Thus, it is assumed by definition that an honest participant will always send the messages he is supposed to. This hardly seems a realistic assumption, however: in practice, almost all faults may occur by accident, or even worse, as a result of dishonest participants sabotaging honest ones to reveal their secrets!

Galil, Haber and Yung [GaHaYu] propose instead a secret sharing in two levels: first, for each secret bit of a user, the user distributes one "x-or share" to each of the other participants, where the exclusive-or of these x-or shares is the original secret bit. The x-or shares are then verifiably secret shared. Recovery is now possible by reconstructing the x-or shares held by the stopped participant, which has the effect of preserving that participant's privacy.

Our protocol could easily be adapted to include the procedure of [GaHaYu]: In case QRS is used, participant P_i would first commit to his input, and then distribute x-or shares of the factorisation of his modulus to the other participants. If P_i stops working, any majority of users could reconstruct the x-or shares of his factorisation, and use instances of our basic protocol to simulate P_i: open his commitments as needed, etc.

In addition to this, the techniques presented in [Ch2] can be used to add flexibility to the protocol. Here, it is shown how to change the *quorum*, i.e. the number of participants necessary to reconstruct the secret, during the protocol. This can be useful in applications, where it can be assumed that almost all faults occur by accident, because of breakdown of communication lines, etc. In this case, it may be desireable to keep the ratio between the quorum and the number of remaining participants constant.

7.6 The Possibility of Further Generalizations

>From an intuitive point of view, it may seem unnatural that only one participant can be unconditionally protected. Why not try to devise a protocol where any subset of the participants could have this option?

To understand this question better, consider the All-or-nothing-disclosure problem (ANDOS).

In this problem, party A posseses a number of secrets, of which she is willing to disclose exactly one to B. B would like to be able to choose which secret to get, without disclosing to A which secret he is interested in.

Clearly, ANDOS is just a special case of the general computation problem. Furthermore, the folklore of the subject has it that for fundamental information theoretical reasons, a two party ANDOS cannot be implemented such that the secrets of both parties are unconditionally protected. Informally, this is so since if A is unconditionally protected, the messages she sends must contain enough Shannon information to determine exactly one of her secrets. But this must then be the secret B learns, which means that he is anything but unconditionally protected!

Allthough we do not yet have a formal proof that this last claim is true, it certainly seems that there is little hope of acheiving the generalisation described above.

Under different assumptions, however, it *is* possible to get unconditional security for all participants. In [ChCrDa], it is shown how to do this using no cryptographic assumptions, in a model where there exists an unconditionally secure secrecy channel between every pair of participants, and where at least two thirds of the participants will follow the protocol.

On the positive side, it should be noted that in joint work with Claude Crepeau, we have recently shown that our basic protocol can be made to work, based only on the assumption that a one way function exists, and that there is a protocol solving the ANDOS problem: Clearly, bit commitments are possible if one way functions exist, and the comparability property can always be satisfied using the fact that all NP-statements can be proved in zero knowledge (allthough using very inefficient protocols). The idea is now to use a protocol for ANDOS to get something which will replace the blinding property: Notice that when participant P_i blinds other participants' commitments, the basic fact used in the protocol is that P_i knows the x-or sum of the bit contained in the original commitment, and the one contained in the blinded version. It will therefore be sufficient if he can learn a set of these x-or relations by doing an ANDOS protocol with each P_j with $j<i$.

The fact that ANDOS can be based on the existence of a trapdoor one way function [Cr] then implies that our protocol can also be based only on a trapdoor one way function. The resulting protocol would have the same property as the one of [GaHaYu] which is based on the same assumption, namely that only one trapdoor function would have to be generated for each participant, while the solution in [GoMiWi] requires generation of $O(n^2)$ functions.

8. Proving all IP-statements in Zero Knowledge

Below, we will informally describe how to use our protocol to construct a zero knowledge proof system for any language L in IP. By definition of IP, we may assume that we have a proof system for L, i.e. a polynomial time Turing machine V (the verifier), a Turing machine P with unlimited power (the prover), and an interactive protocol that can be executed by P and V. The protocol has the property that given a binary string $x \in L$, the verifier will accept x at the end of the protocol with overwhelming probability, while if x is not in L, x will be rejected with overwhelming probability, even if the prover does not follow the protocol.

In each round of the protocol, one of the machines recieves a message, does some (probabilistic) computation, and outputs another message. One can think of these computations as being described by boolean circuits — in particular the circuits describing the verifiers computations can be at most of polynomial size.

The idea is now to use our computation protocol on each of the circuits specifying the verifier's computations. More specifically, the following is done:

At each point where the prover is about to send a message in the original protocol, it will instead send a collection of bit commitments containing the bits of this message. The parties will now use our computation protocol on the circuit specified by the original proof system to be used by the verifier at this point. This circuit may take as input any message sent so far, but also some secret inputoinflips for examplerom the verifier. The comparability property of the bit commitment schemes will ensure that the same set of messages and input is used consistently throughout. The computation protocol will be executed so that the output of the circuit, i.e. the verifiers next message in the original proof system, is private to the prover, by techniques introduced in the previous section. The prover is now free to read this message and compute a response, perhaps even using its infinite computing power.

The last circuit processed is the one doing the computation that the verifier should use in deciding whether or not to accept the proof in the original proof system. The output of this circuit is opened to the verifier, who can check that indeed it is "accept".

It is now easy to see that, because the computation protocol is correct, the verifier can be convinced that it would have accepted, had it done the original protocol with the prover. Note however, that this makes essential use of the fact that one party—n this case the verifieran be unconditionally protected. This means that, even using its infinite computing power, the prover will not learn anything about the secrets of the verifier and is therefore in essentially the same position as in the original proof system.

On the other hand, since the computation protocol is also secure with respect to the secrets of the prover, this new proof system is zero knowledge: the verifier learns nothing except the fact that $x \in L$verything else is hidden in bit commitments, which are computationally as good as nothing.

All this can be summarised in the following

Theorem 8.1.

Assume that pairs of claw free trapdoor functions exist. Then any language in IP has a zero knowledge proof system.

Proof

The existance of trapdoor one way functions will make our basic computation protocol work with conditional security, which is what we need for the prover. The clawfreeness implies the existance of a bit commitment scheme hiding the bits unconditionally [Ch], which means that the verifier can be protected as required above□

Acknowledgements

We would like to thank Gilles Brassard and Claude Crepeau for many stimulating discussions.

References

[BrCr] Brassard and Crepeau: Zero knowledge simulation of boolean circuits. Proc. of Crypto 86.

[Bl] Blum: Coinflipping by telephone: Protocols for solving impossible problems. Proc. of 24. IEEE CompCon, 1982.

[ChCrDa] Chaum, Damgård and Crepeau: Fundamental primitives for multiparty unconditionally secure protocols. To appear.

[Ch] Chaum: Demonstrating that a public predicate can be satisfied while revealing no information about how. Proc. of Crypto 86.

[Ch2] Chaum: How to keep a secret alive. Proc. of Crypto 84.

[Cr] Crepeau: Equivalence between two flavours of oblivious transfers. To appear in proceedings of Crypto 87.

[GaHaYu] Galil, Haber and Yung: Primitives for Designing Multi-Party Cryptographic Protocols from Specifications. To appear.

[GoVa] Goldreich and Vainish: How to solve any protocol problem: an efficiency improvement. Proc. of Crypto 87.

[GoMiWi] Goldreich, Micali and Wigderson: How to play any mental game, Proc. of STOC 1987.

[GoMiWi2] Goldreich, Micali and Wigderson: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. Proc. of Crypto 86.

[GoMi] Goldwasser and Micali: Probabilistic Encryption. JCSS, vol.28, No.2, April 1984, pp.270-299.

[GoMiRa] Goldwasser, Micali and Rackoff: The knowledge complexity of interactive proof systems. Proc. 17th STOC, 1985.

[GrPe] Peralta and van de Graaf: A simple and efficient protocol to prove the validity of your public key. To appear in proceedings of Crypto 87.

[Y1] Yao: How to generate and exchange secrets. Proc. of 27. FOCS, 1986.

[Y2] Yao: Protocols for secure computations. Proc. of 23. FOCS, 1982.

SOCIETY AND GROUP ORIENTED CRYPTOGRAPHY: A NEW CONCEPT

Yvo Desmedt
Dépt. I.R.O., Université de Montréal
Montréal (Québec), H3C 3J7 Canada

1 Introduction

Messages are frequently addressed to a group of people, *e.g.*, board of directors. Conventional and public key systems (in the sense of Diffie and Hellman [4]) are not adapted when messages are *intended for a group* instead of for an individual. To deeply understand the lack of usefulness of the above cryptosystems in the case that messages are intended for (or are originating from) a group of people, let us now nevertheless attempt to use these systems. When conventional and public key systems are used to protect privacy, the legitimate receiver(s) has (have) to know the secret key to decrypt. This means that, a first solution could be, to send the message to all members of the group, *e.g.*, using their public keys. A second is that the secret key is known to all members and that the message is sent only once. All other solutions using a conventional or public key system, are combinations of the above two solutions. We now explain briefly why these two obvious solutions are not adapted to security needs specific to the protection of information intended for groups.

Both solutions are not adequate when members of the group *have* to share *the same responsibility*. To illustrate this we focus on companies who develop software. It is well known, that sometimes supervisors, with *crucial information* about large software developing projects, leave a company to start their own one. If the group, who is developing the software product, had only one supervisor, then frequently the group has to restart at zero, as a consequence of the lack of the crucial information. To avoid this problem, modern software developing companies use two supervisors, such that projects can continue when one of supervisors leaves [6]. One has however to be certain that both supervisors have the same knowledge about the project. In such case the first solution means sending the messages to both supervisors such that the messages are encrypted with their own different public keys. This solution is unsecure when one of them is able to intercept the message for the other. In other words this solution does not guarantee that both supervisors *have to share* the *same* responsibilities. A possible solution could be to use anti-jamming techniques, however this does not exclude that the other supervisors destroys the messages once they arrived. The second solution does not satisfy because the one who reads the message first, can then destroy it to prevent that the other learns it. So this solution does again not guarantee that responsibilities have to be shared.

The purpose of this paper is to solve the above problem and some other similar problems. Before doing this we will wonder what a group of people is and classify these groups (see Section 2). In the same section we will then wonder what kind of protection needs exist for messages intended for groups, or for the messages originating from groups. Each of the protection needs corresponds with a new open problem. In Section 3 we will describe and extend the cryptographic notion of trust. These new notions of trust will be related with the needs for protection of messages intended for groups. In Section 4 we solve some of the new open problems, others remain unsolved. Before attempting to solve these new open problems we restrict ourselves to avoid impractical solutions as much as possible. We however do not pretend at all that the solutions discussed are very practical.

We finally note that the problems which are discussed in this paper, arose by the author from the lecture of Yao [7]. The relation of our paper with his work [7], will be explained in Section 4.

2 Different kind of groups and their cryptographic needs

In this section we discuss the needs. This does not necessarily means that we know how to implement a system which satisfies the discussed needs. So we are discussing the ideal situation.

We mainly discuss (in this section) the needs of groups related to the security of messages intended for the group. In Section 3 we discuss the needs of groups related to messages which originates from the group.

2.1 Groups in our society

Groups play a crucial role in our modern world. Examples of such groups are: the fire department of a city, the board of directors of a company, the Senate and the Congress of the U.S.A. and similar organizations in democratic and non-democratic countries. Each counsel is also a group of people. Other examples can be found in banks having several services, *e.g.*, the service new accounts, the service loans and the service change. In most companies similar structures exist, a well-known example is the financial department, who pays the salaries.

Remark that all the above groups of people have one common aspect: their existence is more known than the name of the people who are members. Remark that the functional aspect of a group is (or must be) independent of its members. Therefor we will call such a group of people ***a group with anonymous membership***. The word anonymous here has not to be taken strictly. Indeed mostly it is no secret who is a member of the group, but it is not necessary to know it. Letters addressed to the group mostly start with the well-known expressions: "Dear Sirs" and the fact that the letters are treated is (has to be) independent from the members. To be more general and extend the notion of groups with anonymous membership we remark that all officials, who once have had the same duty in one office, form a group of people. This is even true if the present incumbent is the only one still living. All these people form a group in time and the functional aspects of the official is (for the greater part) independent from who is in charge. For this reason most kings start their official documents with: "*We* king of ...". To make a distinction between a group of people consisting with more than one member at the same moment and the other ones, we will refer to the first as *groups with anonymous members*, and we call the second group *officials*.

Not all groups in our society have anonymous membership. Sometimes a group of people designates a group of well-known members, *e.g.*, The Beatles. We will refer to these as *groups with known members*.

Before that we discuss what the cryptographic needs are of these groups, we wonder how groups access information intended for them.

2.2 Access of information by groups

If an encrypted message is intended for a group of people, then there exist several methods to access this information. It is evident that not all these methods satisfy the needs of security. To analyze these needs we assume that some members of the group are competitors and that they try to access the information inappropriately or try to hide some information from their colleagues.

Influenced by the example of two supervisors (see Section 1), one could argue that the best method of access to information by a member of the group, for who the message is intended, is that access is only possible if all members agree to open it (decipher it) at exactly the same moment. To demonstrate that this is possible, we assume for the moment that the message is put in a safe with a secret combination of 128 bits. Each of the supervisors knows only 64 bits of the 128. To open the safe both have to agree to meet each other at the safe and have then to bring in their part of the combination to open it and to read the message. We remark briefly that banks use frequently this precaution with safes. A non-mechanical solution to this problem will be discussed later (see Section 4).

The above solution is however sometimes unsecure. A trivial but very important counter-example is the urgent message which is sent to the fire department: "Fire at location ...". Here each member of the fire department must be able to read the message, even if the other members of the fire department are out. Another example is when a fire is destroying the computer of the software developing group. In that case the message is so urgent, that it does not matter that the other supervisor, who has just a day off, does not read it! *So groups can have different needs at different moments*!

Another need is a consequence of the fact that not all members have the same position in the group. A group may have one (or two) supervisors (or presidents). Messages can be intended for the whole group, but only after that the supervisor has read the message. It is normal that the president of the board of directors is informed first that his company has just received an important contract. It would be strange that all other members know it before the president.

To implement the last need some additional memory is necessary, to remember that the president has read the message.

It is evident that after having discussed the three above examples (message is readable if *all* members agree to decipher at the same moment; message is readable by *each* member separately; *if* the president has read the message, *then* it is readable by all members of the group), many other possibilities exist for access by the members of the group. This is trivial to understand if one takes into consideration that a group of people may correspond with a hierarchical society having one president and two vice-presidents. The best method for access could be that messages are first read by the president, then by the vice-presidents and then by the members.

It is clear from the example of the fire in the software developing company, that methods of access can vary from moment to moment. To study this, we now wonder who decides how the information may be accessed.

2.3 Several methods of access: who decides?

Before discussing who decides how the group will access information, intended for them, let us briefly discuss the aspect of keys which will be used when an outsider sends an encrypted message to the group.

In the case that the group has anonymous members, it is evident that it is impossible to use the public keys of all members. So the group will have a common public key, which we call the *group public key* (remark that we do not yet discuss the deciphering).

Several methods can be used to decide how and in which order that the information can be accessed. The first idea is that the group decides it and publishes a group public key corresponding with their needs. The second solution is that a group publishes different group public keys, *e.g.*, one for normal use and one for emergency. In the case of the supervisors, this means that both have to access at the same moment the messages which are sent using the group public key intended for normal use. When the group public key intended for emergency is used, then both can access the information immediately. Let us now discuss the last solution.

When several different access methods have to be used, it would be impractical to publish as many group public keys as there are *possible* access methods. Indeed there are exponentially many methods of access when the group is large. So if the group decides that access methods will differ frequently, then the ideal situation would be that the sender of the information can add a few bits to the message such that these few bits enable the group to access the information *using the method he has decided and no other one*! This means that the group has only to publish one public key and that it can be used in cases that all members have to access the information simultaneously as well as in case of emergency and all other possible cases (depending of what the sender decides). *We remember that we are speaking in this section about needs, what does not necessarily indicate that we know how to build such a system.*

One need related to group encryption needs still to be discussed.

2.4 Who knows about the decision

We have just discussed who decides what the protection need for the messages is (this means how and in which order that they will be accessed). It is clear that those who have decided, know about their decision. But do others know it too? To illustrate the problem let us fix an example. Suppose that the group decided that the messages has first to be accessed by the supervisor and then afterwards by all other members at the same moment. The group publishes the corresponding public key. The question now is: *does the publication of this key and of the encryption method, reveal what the decision of the group is*? If that would be the case, then everybody knows which hierarchy the group has, or more generally knows which kind of society that corresponds with the group.

The last problem is clearly strongly related to the implementation. Does the implementation reveal how the group will access the information, or does there exist a solution which keeps this information secret?

The author is not able to answer this last question for the moment.

2.5 Members leaving

A major problem which was not yet discussed is the problem of members who are leaving the group. Indeed it would be completely impractical that the group public key has to be modified each time that a member leaves the group, or when a new member joins the group! This is certainly true if the group is large.

So an important need is that the group public key has to be modified only if (*e.g.*) the majority of the initial group members is modified.

3 What is trust

All actual cryptosystems rely at the last end on trust. Indeed if you use a key to send information to somebody (using a conventional or public key system) you have to trust the authenticity of the key you use. Even if you use a secret key which you have agreed with your correspondent when you last met him, you rely on trust. Indeed who guarantees you that the person with who you have spoken is not an impersonator!

Problems as the authenticity of the file of the public keys are frequently solved in cryptography by using a so called *trusted party*, *e.g.*, a notary public. However in our society there exist other forms of trust, *e.g.*, two witnesses. Frequently it is better to use two trusted parties instead of one. But what is the optimal form of trust? Again this depend from application to application. In a bank the clerk can handle small transactions on his own. But in most banks the clerk needs a approval from his supervisor for large transactions. The supervisor in a small branch of the bank needs also an approval for wholesale transactions. This means that for banks the ideal situation would be that clerks can use the cryptosystem of the bank on their own for signing (or confirming) small transaction. But that the same cryptosystem refuses to accept a signature for a large transaction, except when countersigned by the supervisor. This means that for outsiders the bank would have only *one* public key, and everybody can verify a signature made by the bank. However making such a signature can depend from the application.

It is clear that there are as many needs as there are forms of society (hierarchical, partially hierarchical, and so on). So the same remarks can be made as above. For example is it necessary that a customer of the bank knows at what level the difference is between small and medium transaction? Or can cryptosystems be made which keep this secret.

4 Solutions

4.1 Introduction

Proposing a cryptosystem which satisfy all these requirements would probably be a revolution in the cryptographic society. Indeed many applications would exist for cryptosystems which satisfy all the above needs. One of the examples we have already discussed was the signature by bank clerks.

But even if such a cryptosystem would be found, this does not guarantee that it is practical. So the author restricted himself to systems which forbid that a ping-pong protocol is used between sender and receiver. Indeed such systems are in many applications impractical due to the delay of the communication. Indeed nevertheless the fact that electronic mail, used between the U.S.A. and Europe, is mostly faster than normal mail, it is impossible to apply ping-pong protocols. Therefor we have restricted the use of such ping-pong protocol. So we *exclude* the use of a ping-pong protocol *between sender and receivers*, but allow that the receivers can use it among themselves. This last concession does mostly not affect practical implementations if the members of the group are on some Local Area Network.

4.2 Solution based on the difficulty of tampering

It is not difficult to make cryptosystems which satisfy *most* of the above needs if one uses systems based on the difficulty of tampering [3]. The reader can easily figures this out by reading [3]. Indeed these systems simplify many implementations (see [2]).

If one has more faith in cryptosystems which security is mainly only based on mathematics, then the discussed approach has to be rejected.

4.3 Solution based on complexity theory

In this case it is no longer true that the author knows solutions for most needs.

Before we discus some solutions, let us first discuss the context in which the ideas have originated. Yao [7] has recently claimed he has a method such that two people (Alice and Bob) can make a number n which is the product of exactly two primes and such that neither Alice nor Bob know what these primes are, except when they both want to recover at the same moment what these primes are. The author has found the following similar problem. The above n would correspond with the public key of the two people and it would be impossible for both to read, without the collaboration of the other, messages encrypted with that key. In order to do this both would have to collaborate such that after the interaction they both know the message, but not the secret key. This last condition is a consequence of the exclusion of a ping-pong protocol between sender and receivers. Before tackling the problem, the author worked on generalizing the needs of encryption of messages intended for (or originating from) groups. Meanwhile the author has realized that one of the problems can easily be solved by using the well-known Blum protocol [1] and another problem by using [5]. We now briefly discuss these two problems and their solutions.

4.3.1 In the case of known members

Suppose that the members of the group are known, *e.g.*, Alice and Bob, and that an outsider *e.g.*, Brigitte, wants to send a message such that Alice and Bob are unable to read it except when they collaborate and such that *no ping-pong* is allowed between the sender and the receivers.

It is easy to solve the above problem by using [1]. We assume that the discrete log problem and that the factoring problem are difficult and that the Blum protocol is secure. In our solution Brigitte encrypts M such that the ciphertext $C \equiv M^x \pmod{p}$, where p is a standard prime. x corresponds with x_1 concatenated with x_2, where x_1 and x_2 are (save) primes. Brigitte enciphers x_1 and x_2 by using respectively the public key of Alice and Bob. So she obtains $E_A(x_1)$ and $E_B(x_2)$, where the index A and B correspond with the public key of Alice and Bob. Then Brigitte signs the complete message consisting in $(M^x, E_A(x_1), E_B(x_2), x_1 * a_1, x_2 * a_2)$, where a_1 and a_2 are random (safe) primes chosen by Brigitte. This signature operation is extremely important to avoid that Bob modifies the $E_A(x_1)$ (and similar related to Alice and $E_B(x_2)$). Brigitte sends the above signed message to Alice and Bob. Neither Alice nor Bob can decipher the ciphertext, but can recover respectively x_1 and x_2. Alice and Bob then use the Blum protocol (or an improved version) to discover the other x_i. Then both Alice and Bob can calculate $x^{-1} \pmod{p-1}$.

It isn't difficult to make variants of the solution. Such variants can have theoretical advantages, certainly that one could argue that the above x has a special form and could help the cryptanalyst. To avoid this x could be divided into its bits. Each of these bits could correspond with one bit of a different prime. These primes are then multiplied with a_i.

4.3.2 Anonymous members

Let us describe the specifications of the necessary cryptosystem. An outsider is sending a message to the group, using the group public key. The group decided that the only possible access to the message is that *if all members of the group are honest*, then access is only possible if all members are willing to read simultaneously. The solution has to take into consideration: that members are leaving and new ones are coming, that if one member dies (or leaves), then the group must be able to continue with the successor (who was unknown before). To realize this last condition the solution may allow that if more than 50% of the members (or ex-members) of the group are becoming dishonest, then this subgroup can do a coup to take over the power of the other members.

The solution is nothing else than an application of [5]. Assume that the group contains always m members (not necessarily the same). First of all the members assume that only u members will become dishonest (where $u/m < 1/2$), while they are a member of the group. All members agree on the public key (or better probabilistic public key) system that will be used. Each member of the group (*e.g.*, member i) chooses a random secret value x_i and large enough. The members agree how they will make the group public key starting from the x_i. In the case of the RSA the public key corresponds with the so called e and n. So for the last case they agree on some Boolean functions, which they will use to go from the x_i to the public key. In order to do so and keeping the x_i secret, they use [5]. The members of the group also discussed how to use these x_i to decipher each message. This means, in the case of the RSA, that they have described how the x_i will be used to perform the operation $M^d \pmod{n}$, where d and n are in fact functions of the x_i. *It is extremely important to remark here that the secret key itself will never be calculated by the group.* So, this means that each time that an outsider has sent a message, the group has to use their secret x_i to finally find the message, certainly not the secret key! So each time that a message arrives, the group applies [5]. So remark that the protocol of [5] is used $k+1$ times, where k is the number of messages received and where 1 corresponds with the use of the protocol for the calculation of the group public key.

When a member leaves the group, one asks him to reveal his secret to his successor. The successor tests then his x_i. This test can be set-up by encrypting a random message and by testing that the group can decrypt it with the value know to the successor, or a zero-knowledge test can be done. In fact in their idea [5] each x_i was encrypted, so that a correct x_i can be checked. If the leaving member refuses to give his old x_i then the other members of the group can use a protocol of [5] to recover the x_i and give it to the new member (they could also do this in a way that nobody else of the members will know what the x_i is). Suppose that there now exist y (ex-members), such that $(y+u)/m$ approaches 1/2, then it is time for the group to make a new public key and announce it.

The same ideas can be used when the group is willing to make signatures, which have to be signed by at least 50% of the members to be valid. (To be correct the word members has to be replaced by: members or ex-members. If we assume again that only u members will become dishonnest a similar approach as just explained is possible).

5 Conclusion

The needs of encryption systems intended for groups has been analyzed. Solutions were adapted from the literature to solve some of the needs. Practical concerns as limitation of ping-pong were taken into consideration. Nevertheless, these solutions are still theoretical, but show that solutions exist for some of the above needs. However it would probably be worse if a ping-pong protocol would have been used between sender and receiver.

The needs of groups are certainly worth to be further analyzed and cryptosystems worked out, which satisfies these needs.

Acknowledgement

The author wants to thank Claude Crepeau (MIT) for having informed the author of the existence of [5] and for the fruitful discussions related to Section 4.3.2 of this paper and [5].

References

[1] M. Blum. How to exhange (secret) keys. *ACM Trans. on Computer Systems*, 1(2):175–193, May 1983.

[2] G. Davida and B. Matt. Arbitration in tamper proof systems. Presented at the same conference (Crypto'87).

[3] Y. Desmedt and J.-J. Quisquater. Public key systems based on the difficulty of tampering (Is there a difference between DES and RSA?). Presented at CRYPTO'86, Santa Barbara, California, U. S. A., August 11–15, 1986, extended abstract will appear in Advances in Cryptology, Proc. of Crypto'86, Lecture Notes in Computer Science, Springer–Verlag, 1987.

[4] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT–22(6):644–654, November 1976.

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth ACM Symp. Theory of Computing, STOC*, pages 218 – 229, May 25–27, 1987.

[6] G. M. Schneider and S. C. Bruell. *Advanced programming and problem solving with Pascal.* Wiley, N.Y., second edition, 1987.

[7] A. C. Yao. How to generate and exchange secrets. In *The Computer Society of IEEE, 27th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 162–167, IEEE Computer Society Press, 1986. Toronto, Ontario, Canada, October 27–29, 1986.

A Simple and Secure Way to Show the Validity of Your Public Key.

Jeroen van de Graaf
Centrum voor Wiskunde en Informatica
Amsterdam

René Peralta
Facultad de Matemáticas
Universidad Católica de Chile

ABSTRACT

We present a protocol for convincing an opponent that an integer N is of the form P^rQ^s, with P and Q primes congruent to 3 modulo 4 and with r and s odd. Our protocol is provably secure in the sense that it does not reveal the factors of N. The protocol is very fast and therefore can be used in practice.

1. Introduction.

Many protocols in the literature assume that the parties involved have previously agreed that their respective public keys are Blum integers. That is, composites of the form $N = P^rQ^s$, with P and Q primes congruent to 3 modulo 4 and with r and s odd. This agreement must of course be achieved through a protocol which does not compromise the secrecy of the keys.

Blum integers are characterized by the following three conditions:

i) $N \equiv 1 \bmod 4$.

ii) There exists a quadratic residue modulo N with square roots with opposite Jacobi symbol.

iii) N has at most two prime factors.

There is no known efficient algorithm to verify conditions ii) and iii). Interactive protocols are used for this purpose: A convinces B that ii) and iii) hold. The protocol for verifying condition ii), due to Manuel Blum [Blum82], requires the interchange of roughly 100 integers modulo N, and thus is very fast. The bottleneck in the published protocols (see [BKP85] [GHY85]) is in showing that N has exactly two distinct prime factors. The protocols are based on an observation due to Adleman. He suggested using the fact that if N has exactly two prime factors then 1/4 of the elements of $\mathbf{Z}_N^*$ are quadratic residues. If N has more than two prime factors then at most 1/8 of the elements of $\mathbf{Z}_N^*$ are quadratic residues. Thus, a binomial experiment can be used to distinguish between the two cases. Ommiting details, the standard solution is to jointly generate M random numbers in $\mathbf{Z}_N^*$ with Jacobi symbol +1 and have the owner of the public key N produce square roots modulo N for approximately half of the numbers. In [BKP85] it is shown that the error probability is minimized by having the prover show square roots for a fraction $(\sqrt{21} - 1)/20$ of the M numbers, giving a probability of error asymptotically bounded by $e^{-(M/75)}$ above and $e^{-(M/74)}$ below. Thus M must be in the thousands in order to achieve truly negligible probability of error. An additional undesirable property of this solution is that the error is two-sided. It is possible that A may fail to convince B that N has exactly two prime factors when in fact it does, and it is possible that A can convince B that N has exactly two prime factors when in fact it doesn't.

We present a much faster protocol that solves this problem and which has only one-sided error probability : with exponentially small probability, A can convince B that N has exactly two prime factors when in fact it has more.

We assume the existence of a mutually trusted source of random bits. This imposes no restriction on our protocol since any of a number of cryptographic techniques can be used to generate mutually trusted random bits.

In the next section we recall the number theoretic definitions and theorems needed for the protocol. Section 3 gives the actual protocol, together with a proofs of correctness and security.

2. Number Theoretic Background.

We denote by $\mathbf{Z}_N^*(+1)$ the set of elements in $\mathbf{Z}_N^*$ with Jacobi symbol +1. If N is a Blum integer then $N \equiv 1 \bmod 4$. This implies the Jacobi symbol modulo N of -1 is $+1$, a fact we will use. We assume B checks that $N \equiv 1 \bmod 4$, not a square and not a power of a prime.

The set of n-tuples $S_n = [\{1,-1\}^n]$ endowed with ordinary component-wise integer multiplication is a group with identity $\mathbf{1} = (1, \ldots ,1)$. Let $P_1^{r1}P_2^{r2}\cdots P_n^{rn}$ be the factorization of N with the P_i's distinct primes. We define the function $\sigma_N : \mathbf{Z}_N^* \longrightarrow S_n$ as follows : the i'th component of $\sigma_N(x)$ is 1 if x is a quadratic residue modulo P_i and -1 otherwise. Notice that σ_N is a homomorphism with kernel the quadratic residues in $\mathbf{Z}_N^*$. If P is a prime or the power of a prime then exactly half the elements of $\mathbf{Z}_P^*$ are quadratic residues. Thus, if x is random in $\mathbf{Z}_N^*$ then, by the Chinese Remainder Theorem, $\sigma_N(x)$ is a random element of S_n. Notice also that if N is a Blum integer then $\sigma_N(-1) = (-1,-1)$.

If the prime powers appearing in the factorization of N all have odd exponents then we say N is **free of squares**. Let $N = VW$ with W the part of N which is free of squares. That is $(V,W) = 1$, V is a square, and W is free of squares. Notice that $W > 1$ by assumption. Since V and N are congruent to 1 mod 4, it follows that W is congruent to 1 mod 4. If x is a random element in $\mathbf{Z}_N^*(+1)$ then, by the Chinese Remainder Theorem, x mod V is a random element in $\mathbf{Z}_V^*$ (provided $V > 1$) and x mod W is a random element in $\mathbf{Z}_W^*(+1)$. From now on we denote by w the number of distinct prime factors of W and by v the number of distinct prime factors of V. Since $W \equiv 1 \bmod 4$ and free of squares, it follows that an even number of prime factors of W is congruent to 3 mod 4. This implies that $\sigma_W(-1)$ has an even (possibly 0) number of negative entries. Another number-theory fact we will use is that a random number x in $\mathbf{Z}_W^*(+1)$ has probability $(1/2)^{w-1}$ of being a quadratic residue. Similarly, a random number x in $\mathbf{Z}_V^*$ has probability $(1/2)^v$ of being a quadratic residue. It follows, by the Chinese Remainder Theorem, that a random number x in $\mathbf{Z}_N^*(+1)$ has probability $(1/2)^{v+w-1}$ of being a quadratic residue. If x is random in $\mathbf{Z}_N^*(+1)$ and $-1 \in \mathbf{Z}_N^*(+1)$ then $-x$ is also a random element in $\mathbf{Z}_N^*(+1)$. Thus the probability that at least one of x or -x is a quadratic residue is less than or equal to $2(1/2)^{v+w-1} = (1/2)^{v+w-2}$ We have shown the following theorem:

Theorem 1: Suppose $N \equiv 1$ modulo 4 and has more than two distinct prime factors. If x is a random element in $\mathbf{Z}_N^*(+1)$ then the probability that at least one of x or -x is a quadratic residue modulo N is less than or equal to (1/2).

Another class of numbers will appear throughout this paper. These are composites of the form $N = P^{2r}Q^s$ with P and Q prime, $P \equiv 3 \bmod 4$, $Q \equiv 1 \bmod 4$, and s odd. For lack of a better name we call these numbers "class II" composites.

Theorem 2: Suppose N is a Blum integer or of class II. Let x be a random number in $\mathbf{Z}_N^*(+1)$. Then either x or -x is a quadratic residue modulo N.

Proof: If N is a Blum integer then $\sigma_N(x) \in \{(1,1),(-1,-1)\}$. Thus x is not a quadratic residue if and only if $\sigma_N(x) = (-1,-1)$. But then $\sigma_N(-x) = \sigma_N(-1)\,\sigma_N(x) = (-1,-1)\,(-1,-1) = (1,1)$ and so -x is a quadratic residue.

If N is of class II then $\sigma_N(-1) = (-1,1)$ and $\sigma_N(x) \in \{(1,1),(-1,1)\}$. Thus if x is not a quadratic residue then $\sigma_N(x) = (-1,1)$ and so $\sigma_N(-x) = \sigma_N(-1)\,\sigma_N(x) = (-1,1)(-1,1) = (1,1)$. □

Theorem 3 : Assume $N \equiv 1 \bmod 4$, N is not a square and not a power of a prime. Let x be a random element in $\mathbf{Z}_N^*(+1)$. If N is not a Blum integer and not a class II composite, then the probability ρ that one or both of x and -x are quadratic residues modulo N is less than or equal to (1/2).

Proof : If N has more than 2 distinct prime factors then, by theorem 1, $\rho \le (1/2)$. So we may assume N has exactly 2 prime factors P and Q.

If $P \equiv Q \equiv 3 \bmod 4$ with $N = P^rQ^s$ then, since $N \equiv 1 \bmod 4$, r and s must have the same parity. Since N is not a square it follows that r and s are odd and therefore N is a Blum integer.

If $P \equiv 3 \bmod 4$ and $Q \equiv 1 \bmod 4$ then N must be of the form $P^{2r}Q^s$ with s odd (since N is not a square and is congruent to 1 mod 4) and therefore is of class II.

The only remaining case is when $P \equiv Q \equiv 1 \bmod 4$. In this case x is a quadratic residue if and only if $-x$ is a quadratic residue. If N is of the form $P^{2r}Q^{s}$ with s odd then $\rho = (1/2)$ because x must be a quadratic residue modulo Q^{s} but is random modulo P^{2r}. Finally, if N is of the form $P^{r}Q^{s}$ with r and s odd, then $\sigma(x)$ is either $(1,1)$ or $(-1,-1)$, each with equal probability. Therefore in this last case ρ is also $(1/2)$.□

We will also need the following facts:

__**Lemma 1**__ : If $N = P^{r}Q^{s}$ is a Blum integer, x a quadratic residue modulo N, and b is 1 or -1, then x has a square root modulo N with Jacobi symbol b.

__**Proof**__. By the Chinese Remainder theorem, $\mathbf{Z}_N$ is isomorphic to $\mathbf{Z}_{P^r} \times \mathbf{Z}_{Q^s}$. Let ψ be the isomorphism mapping $\mathbf{Z}_N$ to $\mathbf{Z}_{P^r} \times \mathbf{Z}_{Q^s}$. Let u be a square root of x modulo N, and let $\psi(u) = (\alpha, \beta)$. Then $v = \psi^{-1}((-\alpha, \beta))$ is also a square root of x modulo N. Recall that the Jacobi symbol of u modulo N is equal to the Jacobi symbol of α modulo P^{r} multiplied by the Jacobi symbol of β modulo Q^{s}. Since $P^{r} \equiv 3 \bmod 4$, the Jacobi symbol of α and $-\alpha$ modulo P^{r} have opposite sign. Therefore u and v have opposite Jacobi symbol.□

__**Lemma 2**__ : Suppose $N = P^{2r}Q^{s}$ is of class II. Let x be a quadratic residue modulo N. Then all square roots of x modulo N have the same Jacobi symbol.

__**Proof**__. Using the notation of lemma 1 the four square roots of x are $\psi^{-1}((\pm\alpha, \pm\beta))$ for some α and β. Now $+\alpha$ and $-\alpha$ have Jacobi symbol 1 mod P^{2r}, and $+\beta$ and $-\beta$ have the same Jacobi symbol modulo Q^{s} since $Q \equiv 1 \bmod 4$. Thus all square roots of x have the same Jacobi symbol.□

__3. How to Convince an Opponent that N is a Blum Integer.__

Assume N is not a square, not the power of a prime, and is congruent to 1 mod 4. The following protocol will convince B that N is a Blum integer.

PROTOCOL :

(1) A and B use the mutually trusted source of randomness to obtain 100 random numbers $\{x_i : i = 1, \ldots, 100\}$ in $\mathbf{Z}_N^*(+1)$ and 100 random signs $\{b_i : i = 1, \ldots, 100\}$ with $b_i \in \{1,-1\}$.

(2) for i = 1 to 100 A displays a square root r_i of x_i or of $-x_i$ modulo N with Jacobi symbol equal to b_i.

Proof of correctness: If N is a Blum integer then, by theorem 2 and lemma 1, A can produce all the square roots required at step (2). If N is not a Blum integer or of class II then, by theorem 3, the probability that A can produce all the square roots required at step (2) is at most $(1/2)^{100}$. But if N is of class II then, by lemma 2, the probability that A can produce all the roots with the required Jacobi symbol is $(1/2)^{100}$ Thus, unless N is a Blum integer, A will get caught cheating with probability $1 - (1/2)^{100}$.□

Proof of security: We must show that if N is a Blum integer then no information is released by A other than this fact. Notice that B simply observes the mutually trusted source of random bits and A's messages. In the terminology of [CEGP86], this is a verifier-passive protocol. It is shown in [CEGP86] that to prove security of verifier-passive protocols we only need to produce a machine S, with input N a Blum integer, which can generate random bits and simulated messages which have the same joint probability distribution as the mutually trusted random bits and the messages sent by A. The simulator S can be constructed as follows:

PROGRAM FOR SIMULATOR S.

i) Generate 100 random elements $r_i \in \mathbf{Z}_N^*$ $\{i = 1, \ldots, 100\}$.

ii) Let b_i be the Jacobi symbol of r_i modulo N.

iii) For each i let $x_i = r_i^2 \bmod N$ or $x_i = -(r_i^2) \bmod N$ with equal probability.

It is a trivial matter to check that the r_i's, the b_i's, and the x_i's have the same joint probability distribution as those generated by A if A is honest.$\square$

Acknowledgements :

We thank Ivan Damgård for helping with an early draft of this paper.

References.

[BKP85] - Berger,Kannan,Peralta, "A Framework For The Study Of Cryptographic Protocols". Proceedings of Crypto85.

[Blum82] - Manuel Blum, "Coin Flipping By Phone". COMPCON. IEEE, February 1982.

[CEGP86] - Chaum, Evertse, van de Graaf, Peralta, "Demonstrating Possession of a Discrete Logarithm Without Revealing It". Proceedings of Crypto86.

[GHY85] - Galil,Haber, Yung, "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems". 26th. FOCS, 1985.

Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model

(Extended Abstract)

Zvi Galil[1, 2, 3] Stuart Haber[1, 3] Moti Yung[1, 3, 4]

[1] Department of Computer Science, Columbia University
[2] Department of Computer Science, Tel Aviv University

We give a general procedure for designing correct, secure, and fault-tolerant cryptographic protocols for many parties, thus enlarging the domain of tasks that can be performed efficiently by cryptographic means. We model the most general sort of feasible adversarial behavior, and describe fault-recovery procedures that can tolerate it. Our constructions minimize the use of cryptographic resources. By applying the complexity-theoretic approach to knowledge, we are able to measure and control the computational knowledge released to the various users, as well as its temporal availability.

[3] Supported in part by NSF grants MCS-8303139 and DCR-8511713.
[4] Supported in part by an IBM graduate fellowship.

Summary

An important area of research in cryptography is the design of protocols for carrying on certain transactions in a communications network, such as playing poker or holding an election. Many of the protocols proposed in this area have required the expensive on-line generation of a large number of new keys. On the other hand, fundamental research in the traditional problems of cryptography, such as encryption and authentication, has developed the *public-key model*, in which each user has a single validated public key. This model is appropriate to those situations in which generation and validation of new keys is very costly or is otherwise limited. Procedures proposed for this model must preserve the security of the keys. An important question is whether flexible protocol design for a wide variety of problems is possible within the public-key model, so that the expense of generating new keys can be minimized.

We identify a broad class of multi-party *cryptographic computation problems*, including problems subject to the partial-information constraints of the "mental games" of Goldreich, Micali, and Wigderson [24]. In order to solve any cryptographic computation problem, we develop new techniques that can be implemented efficiently in the public-key model. By contrast, the constructions of [42] and [24], which were the first to realize general protocol design in the sense of this paper, require on-line generation of many cryptographic keys. We also consider the the temporal constraint that certain computations occur simultaneously. We formalize this as the problem of multi-party *synchronous computation* and give a solution; it was only recently solved by Yao for the case of two parties [42]. Our tools are *minimum-knowledge* protocols, assuring not only the privacy and synchrony of their computational results but also the security of the users' cryptographic keys.

Much current research has been devoted to fault-tolerance in distributed computation, especially in the cryptographic context. None of the fault-tolerance models presented so far has adequately captured the tradeoff between fault-recovery capabilities and maintenance of security; nor do they describe computation in an unreliable communications environment. We introduce a new fault model that allows a more realistic analysis of faulty behavior. We show how to automatically augment any protocol with procedures for recovery from different kinds of faults. These fault-recovery procedures are implemented in the public-key model, and they enable secure recovery from violation failures in such a way as to preserve the security and privacy of all users, including failing processors.

1. Introduction

Recent work in theoretical cryptography has made great progress toward devising general procedures for solving protocol problems [40, 23, 42, 24]. On the other hand, fundamental research in the traditional problems of cryptography, such as encryption and authentication, has developed the *public-key model*, in which each user has a validated public encryption key and a corresponding private decryption key. In designing a public-key system, the cryptographer must be concerned with the security of encrypted messages, as well as the security of users' keys. For example, recent work on probabilistic encryption schemes [25, 7], as well as the work of Goldwasser, Micali, and Rackoff on "zero-knowledge" interactions [26], were motivated by these security issues. In the public-key model, there is an initialization stage during which each user generates a pair of encryption and decryption keys, announces his public key, and validates it by proving that it was properly chosen. Thereafter, as many cryptographic procedures as possible should be carried out using these keys; the procedures performed by the system must guarantee the security of these keys throughout its lifetime.

It can be quite expensive to generate and validate cryptographic keys. First, if good keys occur sparsely in the set of strings of each length, then generating one at random takes a long time. Second, in certain situations it may not be desirable or possible for every processor in a network to have to generate its own key; for example, some of the processors may be independent robots that are provided with built-in keys. Third, as shown recently by Chor and Rabin [12], the process by which a group of users validate their public keys can be the bottleneck in a multi-party protocol. In light of this, it is inconvenient that recent developments in cryptographic techniques for general design of protocols have made extensive use of on-line generation of encryption keys; over and over again, new keys are generated, used once, and then discarded [13, 23, 42, 24]. While this method allows one to prove the cryptographic security of the proposed procedures, it is wasteful of computational resources. Therefore, for both the theoretical and the practical cryptographer, it is important to know how much can be achieved employing only the users' public keys, and exactly when it is necessary to generate new cryptographic keys. The present work identifies a broad class of multi-party cryptographic problems that can be solved within the public-key model, minimizing the cryptographic resources used in solving them.

Cryptographic protocols are used to solve problems that involve hiding information [37, 25, 14, 5, 20, 19, 40, 24] and forcing certain computations to occur simultaneously [32, 9, 39, 28, 4, 17, 42]. The requirements that individual votes in an election protocol remain private, or that cards drawn from the deck in a poker protocol remain private, are examples of partial-information constraints, while exchange of secrets and signing of contracts are examples of problems subject to temporal constraints. We identify a broad class of multi-party problems called *cryptographic computation problems*. This class includes the "mental games" of Goldreich, Micali, and Wigderson [24], which are computations subject to certain partial-information constraints, as well as problems subject to temporal constraints of synchrony. We define multi-party *synchronous computation* and give a general solution for any computational problem; it was recently solved by Yao for the case of two parties [42].

By a "general solution" to a class of cryptographic computation problems, we mean the following. Given a formal specification of one of these problems, we provide an automatic translation into a multi-party protocol which is correct, and which satisfies the given partial-information and temporal constraints. In order to do this efficiently, we develop new *encryption-based* techniques. Our constructions are *minimum-knowledge* protocols that can be implemented efficiently in the public-key model, using public keys that may be based on any family of one-way trapdoor functions. For certain synchronous computation problems, the users may need to cooperate to generate a single additional cryptographic key; for all other cryptographic computation problems, our constructions use only the originally announced public keys of the system. Our minimum-knowledge procedures assure not only the privacy and synchrony of their computational results but also the security of the users' public keys, even after the execution of polynomially many protocols.

If we assume that the Diffie-Hellman key-exchange protocol (based on the discrete logarithm) is secure [16], then we can implement our constructions by simulating the public-key model with only *one* key in the entire network.

Here is an example of the sort of problem for which our techniques can construct a protocol solution (a variation on Yao's millionaires problem [40]): Suppose that a large group of millionaires wish to compute the subset consisting of the one hundred wealthiest among them; the one hundred names that make up this club should only be known to its members and not to the others, and the actual worth of each millionaire should remain secret to everyone. Furthermore, the members of this secret club also wish to generate a common secret encryption key while keeping the anonymity of its owners.

Continuing our study of cryptographic computation, we investigate *fault-tolerance* in the context of cryptographic protocol design. We begin by examining recent work in this area, and observe that none of the models presented so far adequately captures the consequences of users' faulty behavior [13, 23, 24]. The difficulty seems to be that previous models use the notion of faulty behavior that arose in research on the problem of Byzantine agreement. In that domain, a faulty user may attempt to frustrate correct performance by sending different messages to different users, while remaining undetected. In cryptographic protocols, on the other hand, encrypted messages are assumed to be available to all parties. New developments of cryptographic techniques, such as the fact that every NP language has a zero-knowledge interactive proof-system [23], make it possible to design *validated* protocols in which each message is accompanied by an interactive proof that it has indeed been computed correctly. During the execution of a validated protocol, "Byzantine" behavior will be detected (with overwhelming probability) if it is present; thus, it is no longer necessarily the major concern of the protocol designer.

The previous models assumed that all violations, including stop failures, are performed by a malicious adversary; thus the fault-recovery procedures were designed so as to compromise the security of the (presumed) violator --- in fact, to make him give up his identity entirely. This has the paradoxical effect of turning the honest participants into compromisers. One objection to modeling protocols with such a tradeoff between user security and fault-recovery procedures is that these procedures may encourage a real-world adversary, who desires to compromise another user's security, to do this by forcing him to commit a fault (for example by cutting his communication lines, causing a stop-failure). A related objection is that this approach is unsuitable for describing an unreliable communications environment.

We introduce a new fault model for cryptographic protocols. Our model allows a more realistic analysis of faulty behavior, by introducing the cryptographic adversary as a combination of a *compromiser* and a *violator*. We show how to automatically augment any protocol with procedures for recovery from faults. These fault-recovery procedures make it possible to continue a protocol in the presence of faults as long as there is an honest majority, without changing either the distribution of results computed by the protocol, their privacy, or their synchrony. Furthermore, they enable secure recovery from failures in such a way as to preserve the security and privacy of all users, including failing processors. An important contribution of our new recovery procedures is that they eliminate the security-recovery tradeoff of previous work.

In addition, we present a recovery procedure that enables a violator that stops for a while in the middle of a protocol execution to *rejoin* the protocol later, without disturbing the protocol's computational results or their privacy or synchrony constraints. All of our fault-recovery procedures are implemented in the public-key model.

The multi-party computation problems that we discuss in this paper would all be easy to solve if there were a trusted party among the users. This party could receive each user's inputs (encrypted, if necessary), compute the outputs according to the problem specification, and distribute the outputs (once again encrypted, if necessary) to the appropriate users in such a way as to satisfy the given cryptographic constraints, exactly as desired. Because all the computation would be performed by the ideal trusted party, it would be easy to handle faulty processors. Our goal here, both in our work on techniques for cryptographic computation and in our work on fault-tolerance, is to use the complexity-theoretic approach to knowledge in order to design protocols that achieve the same ends in a distributed fashion, in the absence of such a reliable centralized computer.

To summarize, the combination of the results presented here gives a general technique for solving a large class of distributed problems subject to any combination of partial-information and synchrony constraints, using as few cryptographic resources as possible, and with a secure mechanism for recovery from faults. This suggests what might be called the "trusted-party methodology" of cryptographic protocol design: First design a protocol for the idealized situation in which a single trusted party can easily satisfy all the constraints of the problem, and then replace the trusted party with a cryptographic computation that simulates it as efficiently as possible. The work of [42, 24] can also be viewed in this way.

In the next section, we give the definitions we will use. In section 3 we describe our constructions for solving cryptographic computation problems, and in section 4 we present our new fault model and describe our fault-recovery procedures. Some of the technical details of our constructions are in the appendix.

2. Background and Definitions

2.1. The Public-Key Model

In order to study multi-party protocols, we formalize the interacting parties as a system of *communicating probabilistic Turing machines* [26, 20, 14, 13, 5, 24]. Every machine has a private input tape and a private output tape, a tape of random bits, a working tape, and one public communication tape for each of the other machines. In addition, in order to model the fact that the system is not memory-less, we assume that each machine has a private history tape on which it records its computation and its communication activities. There is a global clock, and the protocol proceeds in *rounds*; during each round every machine reads its communication tapes, performs some polynomial-time computation, perhaps using some random bits, and then may write messages on the communication tapes. There is also a global parameter giving the size of the computations to be performed and the size of cryptographic keys (i.e., the security parameter). We note that the communication scenario described here is a design model that can be implemented (using an agreement protocol) on more general point-to-point networks.

The system has an initialization stage during which each machine M_i $(i=1 \ldots n)$ announces its probabilistic encryption key E_i; it is able to decrypt messages encoded using this key. (This is the setting for the original public-key cryptosystems of [16, 35, 34], which used deterministic encryption functions, and of [25], which introduced probabilistic encryption.) We assume that the keys are based on one-way trapdoor functions drawn from the same distribution, and that these functions are uniformly intractable to invert. In addition, the initialization protocol includes a validation of the keys [12]. We will call this the ***public-key model***; the model can be implemented under the general assumption that one-way trapdoor functions exist. Protocols for this model should preserve the security of the cryptographic keys throughout the lifetime of the system.

By an n-party *protocol* we will mean an n-tuple $M=(M_1, \ldots, M_n)$ of such machines. Given the set of keys $E_1, \ldots, E_n$, the global input and its local input, each machine halts in time polynomial in the global parameter with an output string on its output tape, and a string of global output.

For certain applications we extend the model, and specify that the n machines cooperate in order to generate an additional probabilistic encryption key E_0 in such a way that no machine (in fact, no colluding collection of $n-1$ of the machines) knows the corresponding trapdoor information that enables decryption. (In Section 3.2 below we explain how this is done).

2.2. Computational Problems

We are concerned with computational problems as formalized by Yao [42] in the following way. Let $\Sigma=\{0,1\}$. If k is the global parameter, then we will have inputs $(i_0, i_1, \ldots, i_n)$ distributed according to a probability distribution I_k over $(\Sigma^*)^{n+1}$. A problem is specified by requiring that the outputs $(o_0, o_1, \ldots, o_n)$ be distributed according to another probability distribution $O_k(i_0, i_1, \ldots, i_n)$. For example, the special case in which $o_j=f_j(i_0, i_1, \ldots, i_n)$ with probability 1 describes in this framework the problem of computing $n+1$ specified functions of the inputs. Since we are interested in feasible computation, we assume that both the input distributions $\{I_k\}$ and the output distributions $\{O_k\}$ form polynomial-time computable ensembles [41]. Briefly, we will speak of the *computational problem* $[I_k, O_k]$. Without loss of generality (see [33]), the computational problems are given in the form of a probabilistic circuit that computes the desired input-output distribution.

2.3. Protocols and Their Properties

Our aim is to design an n-party protocol $M = (M_1, \ldots, M_n)$ for a computational problem $[I_k, O_k]$, where $E_1, \ldots, E_n$ are the given cryptographic keys. The protocol realizes $[I_k, O_k]$ in a distributed fashion, where i_0 is the public input, i_j is the input of M_j, and similarly o_0 is the public output and o_j is the output computed by M_j. The protocols we design must preserve the security of the keys.

Distributed solutions to a given problem may be subject to cryptographic constraints. For example, we may require that the computation be performed in such a way that certain information remain hidden. Goldreich, Micali, and Wigderson called such problems *partial-information games* (or "mental games") [24]. In particular, we may introduce privacy constraints on inputs, outputs, and intermediate computational results. For example, this may take the form of an anonymity constraint on the identity of the intended recipient of a computed result. Below we formalize these constraints by defining *minimum-knowledge* protocols.

Another constraint we may impose is that of *synchrony*, the requirement that different parties' outputs be computed simultaneously. We remark that the mathematical study of synchrony was initiated in computer science, particularly in the field of distributed computing. Any multi-party computation can be viewed as a competition in which each participant tries to obtain a result significantly earlier than the others; a synchronous protocol is one that prevents anyone from winning. Yao recently proposed a careful definition of the problem of synchronous computation in the case of two parties (calling it the "fairness" constraint), and gave a general solution. Below we give the first solution for the multi-party case.

We define a *cryptographic computation problem* as a composition of one or more computational problems (of the form $[I_k, O_k]$), subject to any combination of partial-information and synchrony constraints. The problem described above in the introduction is one example. Here are two more:

1. Assume that the users of a cryptographic system, each with a private source of randomness, wish to draw public encryption keys at random from a common distribution, one key per user. The parties must somehow combine their private random bits in order to choose their respective keys; however, each trapdoor decryption key chosen should be known only to its owner.

2. Suppose in addition that each of these n users is the representative of a multi-national corporation, and possesses as input data the annual sales and annual income of the company that he or she represents. While desiring to keep their companies' figures secret, the representatives wish to make a certain economic calculation using the data from all n companies. In order to prevent any one representative from computing this output before the others (and then using it in the stock market), they want everybody to learn the result at the same time.

The first problem combines a privacy constraint on the inputs with a privacy constraint on the n-tuple of private decryption keys; the public output is the list of n encryption keys. The second problem is an example of a sychronous computation with input privacy constraints. The problem in the introduction is an example of a computation with an anonymity constraint.

Next, we sketch our definitions that a protocol is correct, and that it satisfies privacy and synchrony constraints. We will formalize these definitions in the full version of this paper.

The definition of correctness is fairly straightforward. A protocol M is a *correct solution* for the problem $[I_k, O_k]$ if with very high probability its outputs are indeed distributed according to O_k when the input distribution is I_k. It is helpful to think of this as an n-party simulation of the situation in which there is a trusted party that can compute the desired output distribution O_k in a centralized fashion.

During the execution of a protocol M, we will say that a message μ sent by party j is *admissible* if it appears to be appropriate to the specified protocol, i.e. if it seems to be that of the specified machine M_j. (We will formalize this notion of "admissibility" in the full paper.) Temporarily, until we deal with fault-tolerant issues, we assume that a non-admissible message stops the protocol.

Goldreich, Micali, and Wigderson proved that, under the assumption that one-way functions exist, every language in NP has a zero-knowledge protocol for proving membership. This fact has important consequences for the design of protocols [23]. It enables on-line validation that a protocol is being executed correctly. Each message transmission is followed by an interactive proof to all parties that it is indeed a correct message; an incorrect message will be caught with very high probability. This validation can be most efficiently performed by a direct minimum-knowledge simulation of the computation that produced the message, following the construction of [29]. We call such a protocol a *validated* protocol. Our protocols will be validated ones.

A correct validated protocol is one that achieves the computational task that is its goal. Next we deal with controlling the knowledge revealed to the participants in a protocol; we give our definitions that capture those protocols that achieve only the specified task and nothing more.

In order to describe the cryptographic constraints precisely, one might try to give a formal definition of users' changing "knowledge states" during the course of a protocol execution [15]. Instead, we take a computational approach to knowledge and consider certain objects that are theoretically more tractable than knowledge states, namely the sets of strings that may appear on the various tapes of the n Turing machines [26, 20, 42, 23]. We assume that the reader is familiar with the notion of polynomial-time indistinguishable ensembles of strings [25, 41, 26].

In order to define the minimum-knowledge and synchrony properties of a protocol $M=(M_1, \ldots, M_n)$, we must consider the possibility that several of the n parties may *collude* with each other. Colluding parties $j_1, \ldots, j_c$ are able to communicate with each other through private channels --- in particular, they can share their inputs $i_{j_1}, \ldots, i_{j_c}$ and their decryption keys. They may follow any (feasible) programs $M_{j_1}', \ldots, M_{j_c}'$, and the protocol continues as long as the messages they send are admissible. Groups of colluding parties may be created dynamically.

Given a protocol M, consider a set of c colluding parties $j_1, \ldots, j_c$. We desire that they should gain no more information from the messages sent during an execution of the protocol --- e.g. information that can confer a computational advantage in guessing the input or output or compromising the encryption key of another machine that *is* following the protocol --- than they would learn simply by consulting a *result oracle* (which knows *all* the inputs $i_1, \ldots, i_n$ and has access to the colluding machines) to obtain their output values $o_{j_1}, \ldots, o_{j_c}$. We call a protocol *minimum-knowledge with respect to $c(n)$-subsets* if it satisfies this requirement with respect to every possible dynamically chosen subset containing as many as $c(n)$ of the machines. Of course, the strongest possible requirement is that $c(n)=n-1$.

We formalize the definition of the minimum-knowledge property by stipulating that for any subset of possibly colluding machines, there should exist a probabilistic polynomial-time *simulator* S, as follows. Given access to the internal state and all the tapes of the colluders, and allowed queries to the result oracle for the colluding machines' outputs, S produces a simulation of the colluders' *views* of an actual execution of the protocol. (Each machine's view, recorded on its history tape, includes all messages sent during the execution and the random-tape bits it has used in its own computation, as well as any privately computed outputs.) The simulation should be indistinguishable, by any feasible computation that may use the colluders as subroutines, from a transcript of the colluders' views of an actual execution. This definition generalizes the notions of "minimum-knowledge" and "result-indistinguishable" protocols that transfer computational knowledge [26, 20] to the case of n-party distributed computation.

It is a consequence of this definition that if a protocol M is minimum-knowledge, then the security (in the sense of [25]) of the key E_j of any non-colluding machine M_j is not compromised by the protocol.

As in the case of two-party minimum-knowledge protocols [20], it is crucial to our proofs of the theorems in this paper that if two multi-party protocols are both minimum-knowledge, then so is their concatenation. We will give a proof of this concatenation lemma in the full paper.

Next we define synchrony for a protocol M. Once again, consider a set of c colluding parties. We desire that at any moment the colluders can gain no more computational advantage over any other machine in computing the outputs than they already have simply by knowing their inputs. A protocol will be called *synchronous with respect to $c(n)$-subsets* if it satisfies this requirement with respect to every subset containing as many as $c(n)$ of the machines.

More precisely, at any fixed time during the protocol execution, consider the problem of distinguishing which of two given strings (drawn from the space of possible computed outputs defined according to *a priori* knowledge of the inputs) will turn out to be the actual result of the computation. The protocol achieves synchrony if the colluders' best distinguishing probability between two different strings in their result-space is essentially the same as the best distinguishing probability for any non-colluding user.

3. General Protocol Design

In this section we present a general technique for constructing a protocol solution to any cryptographic computation problem in the public-key model. Given a circuit for the computational problem $[I_k, O_k]$ and a set of cryptographic constraints that the solution must satisfy, we construct a protocol solution. The technical tool we introduce is an *encryption-based* circuit simulation, motivated by the recent work of Yao [42] and Goldreich, Micali, and Wigderson [24] on protocol design, and by the security provided by probabilistic cryptosystems [25, 41, 8, 7].

We state here the main result of this section.

Theorem 1: Given a set of polynomially many n-party cryptographic computational problems, there is a correct polynomial-size minimum-knowledge protocol solution. The protocol uses only the public keys $E_1,\ldots,E_n$, if there are no synchrony constraints among the given computational problems.

Our proof is a constructive one, automatically translating the given problems to a solution protocol. Below we present our techniques.

3.1. Two-Party Computation

First we consider the case of two parties, A and B. We are given a circuit that has output distribution $O_k(i_A,i_B)$ when its input (i_A,i_B) is distributed according to I_k. For simplicity, suppose that O_k is the desired output for B. We describe here a minimum-knowledge two-party protocol that simulates the given circuit. We contrast our solution for the public-key model with the protocol solution of Yao [42], which required the on-line generation of $O(C)$ cryptographic keys for a circuit of size C.

Our construction uses probabilistic encryption and cryptographically secure pseudo-random bit generators based on the users' public keys [6, 41, 31]. (See also [30, 10, 8, 2, 7, 38, 21].) If we make the special assumption that the Diffie-Hellman key-exchange protocol (based on the discrete logarithm) is secure [16], then we can also implement all of our constructions by simulating the public-key model with only *one* key in the entire network.

The protocol has several stages. During the first stage one of the parties, the *circuit-constructor A*, prepares an encryption-based circuit-simulation for the use of the other party, the *circuit-evaluator B*. B then verifies that A's construction is as specified in the computation problem. Next, A and B combine their inputs in such a way that they remain secret, and finally B evaluates the output.

A uses his probabilistic encryption key to construct the circuit-simulation, and uses a minimum-knowledge proof to validate the construction. The simulation protocol proceeds by simulating the gates of the given circuit. The protocol consists of three kinds of (simulated) *gates*: input gates, intermediate gates and output gates. Each gate has *entries* for input and output, and a "truth table" that maps the values of the input entries to the output entries. An entry, representing a bit in the computation, consists of two random bit-string *cleartexts*, one for each possible value of the bit; the entry is presented to B as the pair of encoded *ciphertexts* of the two cleartexts. A *valued* entry is one whose cleartexts have low-order bit (for example) equal to the bit represented. An entry may be either *ordered*, in which case the two cleartexts represent 0, 1 (in that order), or *unordered*, in which case they represent either 0, 1 or 1, 0 (at random). An entry may also be *marked*: each of the ciphertexts is tagged with a probabilistic encryption of the bit that it represents.

The inputs to the simulation are the input entries of the input gates. A's input entries, corresponding to the bits of i_A, are unordered and marked, while those of B, corresponding to the bits of i_B, are ordered and unmarked. The truth table of every gate consists of "rows", each of which enables the evaluator to combine one cleartext of each of the gate's input entries in order to obtain the cleartext of an output entry; the rows of the truth table are permuted at random. The output of each input or intermediate gate "connects" to the input of the next gate in the circuit, because the first gate's output entry enables B to decrypt one of the input-entry ciphertexts of the next gate. These intermediate entries of the circuit are unmarked and unordered. The output gates of the circuit have output entries that are valued.

B uses the circuit-simulation in three stages: an input stage, a computation stage, and an output stage. During the input stage B gets his input --- i.e., receives from A the cleartext corresponding to each of his input bits --- by the ***generalized oblivious transfer*** protocol [24]. (This is an extension of the *oblivious transfer* [27, 19] that can be implemented in the public-key model. It provably assures that B receives just one of the two cleartexts, according to his choice, while A does not learn which one is transferred). In case an encryption of the input bit is publicized by B in advance, B also interactively proves that he has received the correct cleartext. Then A does the following for each of his (unordered) input entries: without revealing its tag, he sends B the cleartext that represents the value of his input bit. (Thus B does not learn what this value is.) In case an encryption of the input bit is known, A also proves to B that he sent the correct cleartext.

Once he has a cleartext for each of a gate's input entries, B computes a cleartext for that gate's output entry according to the truth table. (We sketch the details of the truth table in appendix I.) Since this output entry is not ordered, B cannot tell what bit-value this cleartext represents. Using the "connections" between gates, B then continues through the circuit, computing output cleartexts for the intermediate gates and, finally, for the output gates of the circuit. Since these, the output entries of the circuit, are valued, B learns the values of the output bits, i.e. the result of the required computation. If desired --- i.e. if O_k is the desired output for A as well as for B --- B can now tell the result to A, and convince him of the value of each output bit by revealing the corresponding cleartext he computed. The *entire interaction* is minimum-knowledge: neither A nor B learns more about the other's inputs, or about the others' decryption key, than they would if an oracle simply told them the output. (More formally, in the actual computation each party "learns" only random values; given the output by an oracle, each of them could generate a simulated protocol transcript that is polynomial-time indistinguishable from an actual one.)

Further technical details of our construction are described in appendix I.

3.2. Multi-Party Computation

Now we consider the general case of n parties. Given a circuit for an n-party computational problem, the above construction for two parties can be used as a basic step in order to give an n-party minimum-knowledge protocol that simulates the circuit. We contrast our solution for the public-key model with the protocol solution of [24], which required the on-line generation of $O(n^2C)$ cryptographic keys for a circuit of size C.

At any point during the course of the circuit-simulation, each bit b in the simulated computation is shared by the n parties: each party holds a secret share of the global bit, but this share is random and gives no information about the global value. Each gate of the simulated circuit is replaced by an n-party computation whose inputs are the n-tuples of shares of the gate's inputs, and whose outputs form a similar distribution of n shares of the gate's output. This n-party computation consists of $n(n-1)$ two-party interactions as above.

The construction of [24] uses Barrington's encoding of circuits with elements of the permutation group

S_5 [3]. Each user's share of a globally distributed bit is a permutation, and two-party circuits of Yao [42] are used in order to "compute" with these permutations. In joint work with Silvio Micali, we simplify the representation and manipulation of distributed shares. The simplification is that each user's share of b is simply a random bit b_j satisfying $b = \sum_{j=1}^{n} b_j \pmod 2$. We show how to compute with these shares, by making some simple bit-computations, in appendix II.

Continuing the work described above, several authors have recently proposed protocols for different sorts of circuit simulation, basing their cryptographic security on certain complexity-theoretic assumptions [22, 1, 11].

3.3. Synchronous Computations

Given a circuit C for an n-party computational problem, we give a protocol solution that satisfies the synchrony constraint.

Yao gave a protocol for synchronous two-party computation [42]. As in the construction of section 3.1 (which was motivated by his work), the two parties have asymmetric roles in his protocol; one party "constructs" the circuit simulation, while the other party "evaluates" it. In order that the computation be synchronous, Yao proposes the following: first, the two parties together generate a trapdoor function whose secret trapdoor they do not know; next, they use this function to encrypt their computational results; and finally, they cooperate to recover the trapdoor simultaneously.

In order to implement multi-party synchronous computation, the n parties jointly generate a random cryptographic key E_0 that will be used in order to hide the computational results, in such a way that all n parties must cooperate in order to recover them. At any point in the recovery process, all parties have equal computational advantage in computing the desired result. We describe the details of our solution in the full paper.

4. Fault-Tolerance

When we introduce faults to our model of computation we complicate the protocol-designer's task: users may not follow.their instructions, they may try to extract additional information from the messages sent during the execution, or they may completely stop functioning. Up to this point, we have assumed that all parties behave admissibly, and otherwise the protocol stops; thus the correct completion of the protocols we have described depends on this assumption. Where it is possible, we would like to augment our protocols with the capability to detect and recover from faulty behavior. Recent work [13, 23, 24] has dealt with this problem. In this section we give a careful analysis of faulty behavior in the context of cryptographic protocols, and present new protocols procedures for recovery from faults.

An important tool for fault-tolerant protocol design is that of *verifiable secret-sharing* [13, 36]. A verifiable secret-sharing protocol with parameters (m, t) is a procedure by which a sender can distribute to each of m receivers a *share* of a given secret s, in such a way that it is easy to verify that each share is indeed a proper share, it is infeasible to obtain any information about s from any l of the shares, as long as $l < t$, and it is easy to compute s from any t of the shares. A protocol that achieves this can be given (for any $t \leq n$).

Using verifiable secret sharing, Goldreich, Micali, and Wigderson devised a procedure that transforms any given protocol into a validated protocol tolerating up to $t(n)$ faults, where $t = \lfloor (n-1)/2 \rfloor$ Faulty processors may deviate from their specified algorithms in any arbitrary but feasibly computable way. Here we sketch their transformation. Let $h = n-t$. The transformed protocol begins with a set-up phase during which each processor $(n-1, h)$-verifiably secret-shares its input and its random tape. Until this phase is over, the protocol is vulnerable to faults; the only way to continue the protocol is by restarting the set-up procedure. When a fault is detected during the execution of the transformed protocol, the faulty processor loses its identity completely: the honest users use their secret-shares to reconstruct its input and random tape, and then simulate its role throughout the rest of the protocol. The transformation is *correctness-preserving*, in that honest parties compute the same outputs as in the original protocol; and it is *privacy-preserving*, in that whatever a $t(n)$-bounded adversary can compute in the transformed protocol, he could also have computed in the original one [23, 24]. If the original protocol is minimum-knowledge with respect to c-subsets, for any $c < h$, then so is the transformed protocol.

Notice that in the procedure just outlined, a faulty user is always severely punished; its input is compromised, even if its violation was an unintentional stop-failure caused, for example, by an undelivered message. The procedure is based on the pessimistic assumption that all faults are malicious ones, which only holds in a fully reliable communications environment. The notion of faulty behavior implicit in this assumption coincides with that of research in Byzantine agreement. In that domain faulty behavior is undetected, and messages are private. In cryptographic protocols, on the other hand, encrypted messages are assumed to be available to all parties. Furthermore, during the execution of a validated cryptographic protocol "Byzantine" behavior is detected (with overwhelming probability) whenever it occurs. Since all players know this is true, in practice it may be the case that most faults will be unintentional. Paradoxically, the transformation of Goldreich, Micali, and Wigderson turns the honest majority into compromisers. Here we present a more realistic analysis of possible faulty behaviors, and describe a new fault-recovery procedure that avoids imposing on inadvertently faulty processors the heavy penalty demanded by their procedure.

4.1. New Fault Model

The first question one must ask is when it is possible for a protocol to recover even from a single fault. If the protocol is to continue after a violation, then any recovery procedure must confer on the honest users that detect the faulty processor the power to act on its behalf in the continuation of the protocol. But there are protocols --- e.g. poker-playing, stock transactions --- in which we don't want to recover in this way; even if a user (perhaps inadvertently) times out of today's transactions, he may want to play cards (or buy stocks) tomorrow, using the same secret card-playing (or stock-market) strategy. This strategy is a part of the processor's program that should never be shared with others. It is only in the case of protocols in which each processor's entire program is publicly known that it makes sense to look for recovery procedures. For the rest of this paper, we assume that all protocols are of this form.

We begin by distinguishing between two different sorts of faulty behavior: passive *compromise*, by which we mean the attempt to extract secret information while seeming to act according to the protocol, and active *violation*, intentional or unintentional, of the protocol's instructions, such as by sending erroneous (Byzantine) messages or by stop-failure. We model these two kinds of faults by speaking of an adversary who is able to corrupt a bounded number of the processors. The adversary may, at any time during the execution of our protocol, choose a processor to compromise, until he has chosen $c(n)$ *compromisers*; he has access to the internal state and all the tapes of each of the compromisers, and he may perform any (feasible) computation with this information. Compromising behavior is, by definition, undetectable by an honest user; the most that we may demand is that the protocol be designed so that compromisers cannot gain anything from the information that they share. In fact, a protocol that is minimum-knowledge with respect to $c(n)$-subsets can tolerate a $c(n)$-bounded compromising adversary. The adversary may also choose up to $v(n)$ *violators*; at any point during the execution, he may block the messages sent by any of the violators or over-write them with messages of his choice. From the point of view of an honest processor, violating behavior is of two sorts: stop-failure, which is detectable when the violator times out (according to the global clock) and fails to send an expected message during a round of the protocol; and sending erroneous messages, which will be detected (with very high probability) if our protocol is a validated one. It can happen that a processor can be both a compromiser and a violator; this situation can model what the authors of [24] call a "malicious adversary", which is the strongest form of adversarial behavior. However, we do not assume that every fault must be malicious.

To fix notation, let $h(n) = n - c(n) - v(n)$ denote the minimum number of *honest* processors, those that are neither compromisers nor violators.

4.2. Fault-Recovery Procedures

We suggest a new protocol transformation that fits the more realistic fault model just described. The transformed protocol preserves the privacy of *all* inputs, including those of a violator, and can be implemented in the public-key model. More precisely, the transformation is *discretion-preserving*: during an execution of the transformed protocol an honest user computes no more about the input of any other user, including violators, than he computes in the original protocol. Furthermore, we give a *rejoin procedure* by which a disqualified faulty machine may rejoin the protocol execution in order to obtain its computed outputs, without disturbing the original protocol's computational results or the cryptographic constraints they satisfy.

Here we state the main result of this section.

> **Theorem 2:** Any n-party protocol may be transformed into a validated protocol that is secure against $c(n)$ compromisers and tolerates $v(n)$ (disqualified and rejoining) violators, as long as $v(n)+c(n) \leq n-1$; the transformation is of polynomial cost, and is correctness-preserving, privacy-preserving, and discretion-preserving.

The transformed protocol begins with a *set-up phase*, during which processors' inputs are distributed (by verifiable secret-sharing); in order to qualify for participation in the protocol a machine must take part in this stage. When a violation occurs --- up to $v(n)$ of them are possible --- the set-up procedure restarts. When a violation is detected during the *execution phase*, the violating machine is *disqualified* and the *recovery procedure* is invoked.

Let $M = (M_1, \ldots, M_n)$ be the original protocol. The transformed protocol $M' = (M_1', \ldots, M_n')$ proceeds as follows.

SET-UP PHASE

Each machine M_i', for each bit b of its given input and its generated random tape, chooses bits b_j ($j=1 \ldots n$) at random satisfying $b = \oplus_{j=1}^{n} b_j$, and sends $E_j(b_j)$ to M_j'. (We will call b_j a *piece* of the *distributed* bit b.)

Each machine M_i' follows a verifiable secret-sharing protocol with parameters $(n-1, h)$ to share with all the other machines its piece of every distributed bit.

EXECUTION PHASE

Follow M, validating every message. Whenever M requires a disqualified machine M_j to compute a message μ, the active machines simulate the computation of μ, using the distributed bits of M_j (via a multi-party cryptographic computation protocol as described in Section 3.2).

Upon detection of a violation by machine i, invoke the protocol RECOVER(i).

When disqualified machine M_i' requests to rejoin the execution, invoke the protocol REJOIN(i).

RECOVER(i)

Disqualify machine M_i'. Let j be the first index in $\{i+1 \bmod n, i+2 \bmod n, \ldots\}$ such that machine M_j' is not disqualified.

For each globally distributed bit b, the active machines reconstruct the violator's piece b_i, and machine M_j' updates its piece by setting $b_j := b_j \oplus b_i$)

Note that it is possible to recover in this way, even if another processor should fail during a reconstruction step.

REJOIN(i)

For each globally distributed bit b, and for each machine M_j' that is not disqualified, M_i' chooses a bit r_j at random, and performs a 2-party simulated computation with M_j' that has the effect of setting $b_j := b_j \oplus r_j$; then M_i' sets $b_i := \oplus_j (r_j)$.

Observe that the number of times that the REJOIN procedure may be invoked is bounded by the length of the original protocol, since at least one step must intervene between a disqualification and a rejoining.

We call attention to a trade-off between the maximum number of compromisers and the maximum number of violators that can be handled using these methods. By the minimum-knowledge properties of verifiable secret-sharing, the violators' privacy is preserved as long as $c(n) \leq h(n)-1$. Since h users are needed in order to reconstruct each violator's shares, the maximum number of violators that can be tolerated satisfies $v(n) \leq n-h(n) \leq n-c(n)-1$; in other words, we must have $v(n)+c(n) \leq n-1$. In case there is no distinction made between compromisers and violators, the maximum number of faults is $\lfloor (n-1)/2 \rfloor$.

In the full version of this paper, we will describe a *dynamic recovery* procedure for a model in which there is a failure rate (instead of bounds $c(n)$ and $v(n)$) given as a fault parameter.

5. Conclusions

In this paper we address a number of issues in the design of protocols to solve general multi-party computational problems subject to various cryptographic constraints. First, we present new techniques that enable the automatic translation of a problem specification into a multi-party protocol satisfying any given partial-information and temporal constraints; the resulting protocol can be implemented in the public-key model, requiring the generation of new cryptographic keys only for certain synchronous computation problems. Second, we present new fault-recovery procedures that make it possible to continue a protocol in the presence of faults as long as there is an honest majority, without changing either the distribution of results computed by the protocol or the cryptographic constraints they satisfy, while at the same time preserving the security and privacy of all users, including failing processors.

In both parts of this work we apply the complexity-theoretic approach to knowledge in order to measure and control the computational knowledge released to each of the participants in a protocol. This approach enables us to design protocol procedures that simulate, in a distributed fashion, a central trusted party that assures the correctness, privacy, and synchrony of the results computed. This "trusted-party methodology" of protocol design can be applied, for example, to existing protocols that require a central server (e.g. [14, 18]) in order to make them fully distributed.

Appendix

I. The Basic Two-Party Construction

Here we describe the technical details of a simulated gate in a two-party circuit simulation.

For this description, we implement the operations of probabilistic encryption as follows. The one-way function E is used to encode the cleartext a by choosing a bit-string r at random and computing the ciphertext $x = E(r) \oplus a$. We will call r the *random seed* for the ciphertext x. Without the trapdoor information for E, it is an intractable problem to recover a (or any value that functionally depends on a) from x.

As in Section 3.1, we call the two parties A and B (for Alice and Bob). A is the circuit-constructor and B is the circuit-evaluator.

Instead of giving our construction in complete generality, we will explain the case of an input gate with two input entries that computes i_A OR i_B, where i_A is A's input bit and i_B is B's input bit. The two entries are presented to B as two pairs of ciphertexts, $[x_0, x_1]$ and $[y_0, y_1]$, unordered and ordered, respectively.

Next, A and B perform the generalized oblivious transfer protocol, so that (according to his choice of the bit i_B=0, say) B obliviously receives the random seed s_0 for the ciphertext y_0; he can then decode it and recover the cleartext b_0. A then will send B either r_0, the random seed for x_0, or r_1, the random seed for x_1, depending on whether A's input bit i_A is 0 or 1. Say, B receives r_1; he can decipher x_1 and recover the cleartext a_1.

Note that after these computations it is an intractable problem for B to extract a_0 from x_0, or b_1 from y_1.

Let a_0^L, a_0^R denote the left and right halves of the bit-string a_0, and similarly for the other cleartext strings. Along with its entry ciphertexts, the gate is presented with "instructions" of the following form.

- 1st, 1st : L, L
- 1st, 2nd : R, L
- 2nd, 1st : L, R
- 2nd, 2nd : R, R

What this means is that if the two cleartexts received by B correspond to the first and the first (respectively) of the two pairs of entry ciphertexts, then B should compute the XOR of the left half and the left half (respectively) of the two cleartexts; if they correspond to the first and the second, he should XOR the right half of the first with the left half of the second; and so on. Each instruction corresponds to a line of the gate's truth table, and the lines have been randomly permuted.

The cleartexts have been chosen at random by A so as to satisfy the equations

- $a_0^L \oplus b_0^L = c_0$,
- $a_0^R \oplus b_1^L = c_1$,
- $a_1^L \oplus b_0^R = c_1$, and
- $a_1^R \oplus b_1^R = c_1$,

where c_0 and c_1 are random strings of the appropriate length. (Explicitly, A could build the gate by first choosing the order of the four instructions, then choosing c_0, c_1, a_0, a_1 at random, and finally computing b_0, b_1 according to the equations.)

In our case, B has received a_1 and b_0, so he follows the third instruction and computes c_1. The fact that he also knows the string b_0^L gives him no help in using the first equation to compute c_0, since he has no information about the random string a_0^L. Similarly, since he does not know which of the other three "rows" of the truth table would give him the same result-string c_1 as the row which he did use, he cannot use the string a_1^R to learn anything from the fourth equation. And he knows neither of the bit-strings on the left hand side of the second equation.

Thus, in any execution of the evaluation protocol, B computes a single result-string, and gains no computational advantage in guessing any other result-string; and this is what we need in order to assure the security of intermediate computational results. The result string is used in turn as a random seed, either for an input entry of the intermediate gate to which this gate is connected, or for the simulated circuit's output.

II. Multi-Party Computation

Here we show how n parties can use their shares of globally distributed bits to simulate circuit computations.

Throughout this section, all arithmetic expressions should be interpreted mod 2. At the beginning of the protocol, each machine does the following for each bit b that it needs as input to the circuit simulation: it chooses bits b_j ($j=1 \ldots n$) at random satisfying $b = \sum_{j=1}^{n} b_j$ and sends $E_j(b_j)$ to M_j.

How do n parties "compute" with these shares? Suppose a and b are two shared bits; that is, machine M_i possesses share a_i of bit a and share b_i of bit b. These random shares satisfy the equations $a = \sum_{i=1}^{n} a_i$ and $b = \sum_{i=1}^{n} b_i$.

To simulate a NOT gate, i.e. to simulate the computation $a := 1-a$, one of the machines, say the first, complements its share: $a_1 := 1-a_1$. Now every machine possesses a proper random share of the complementary bit $1-a$. For the simulation of an AND gate, i.e. of the computation $c := ab$, consider the following equations, where each r_{ij} is a bit randomly chosen by M_i:

$$ab = \left(\sum_{i=1}^{n} a_i\right)\left(\sum_{i=1}^{n} b_i\right) = \sum_{1 \le i,j \le n} a_i b_j = \sum_{i=1}^{n} a_i b_i + \sum_{i \ne j} [r_{ij} + (r_{ij} + a_i b_j)].$$

This sum can be rewritten as

$$\sum_{i=1}^{n} \{a_i b_i + \sum_{j \ne i} [r_{ij} + (r_{ji} + a_j b_i)]\} = \sum_{i=1}^{n} c_i ,$$

where $c_i = a_i b_i + \sum_{j \ne i} [r_{ij} + (r_{ji} + a_j b_i)]$ is the share of c that will be computed by M_i at the end of the computation. Each pair of machines, M_i and M_j, carry on two different two-party interactions. In the first of these, M_i acts as the circuit-constructor and M_j acts as the circuit evaluator for a circuit with inputs r_{ij}, a_i from M_i and b_j from M_j, and output $r_{ij}+a_i b_j$ for M_j. Machine M_i holds the value r_{ij} and mcahine M_j holds the value $r_{ij}+a_i b_j$; thus the two machines hold random shares whose sum is equal to $a_i b_j$. In the second of their interactions, they reverse their roles in order to share $a_j b_i$. (Notice that the required two-party computations are very simple.) Each machine M_i takes the sum mod 2 of these $2(n-1)$ output values, and adds $a_i b_i$ to it; the resulting sum c_i is its share of the new globally distributed bit $c=ab$. This concludes the simulation of the AND gate.

The multi-party circuit-simulation just described is minimum-knowledge with respect to $(n-1)$-subsets, and it can realize computations with privacy constraints. At the end of the simulation, the circuit's outputs are "computed" as follows. For each bit b of the public output o_0, each machine M_i announces its share b_i. For each bit b of the private output o_j for machine M_j, every other machine M_i, $i \ne j$, reveals its share b_i.

Acknowledgements

We would like to thank Silvio Micali for introducing us to this area of research and for many helpful discussions, and Cynthia Dwork for her careful critical remarks, as well as the members of the Dipartimento di Informatica ed Applicazioni at the University of Salerno, who were our hosts when we began the research reported here.

References

1. Abadi M., and J. Feigenbaum. A Simple Protocol for Secure Circuit Evaluation. Preprint, 1987.

2. Alexi, W., Chor, B., Goldreich O. and Schnorr C.P. RSA/Rabin Bits are 1/2 + (1/poly(k)) Secure. Proc. 25th FOCS, IEEE, 1984, pp. 449-457.

3. Barrington, D.A. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC_1. 18th STOC, ACM, May, 1986, pp. 1-5.

4. Ben Or, M., O. Goldreich, S. Micali, and R. Rivest. A Fair Protocol for Signing Contracts. Proceedings of ICALP-85, July, 1985, pp. 43-52.

5. Benaloh, J.C. and Yung M. Distributing the Power of a Government to Enhance the Privacy of Voters. Proc. 5th PODC, ACM, 1986, pp. 52-62.

6. Blum, M. and Micali, S. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. Proc. 23rd FOCS, IEEE, 1982, pp. 112-117. Also in: *SIAM Journal on Computing*, November 1984, 850-864.

7. Blum, M. and S. Goldwasser. An Efficient Probabilistic Public-Key Scheme Which Hides All Partial Information. Proceedings of Crypto84, 1985, pp. 289-301.

8. Blum, L., Blum M. and Shub M. Comparison of Two Pseudo-Random Number Generators. Proceedings of Crypto82, August, 1982, pp. 61-78.

9. Blum, M. "How to Exchange (Secret) Keys". *ACM Transactions on Computer Systems 1*, 2 (May 1983), 175-193.

10. Boppana, R.B. and R. Hirschfeld. Pseudorandom Generators and Complexity Classes. Preprint, 1986.

11. Chaum D., I. Damgard, and J. van de Graaf. Multiparty Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output. These proceedings.

12. Chor, B. and Rabin M.O. Achieving Independence in Logarithmic Number of Rounds. 6th PODC, ACM, August, 1987.

13. Chor, B., Goldwasser S., Micali S. and Awerbuch B. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. Proc. 26th FOCS, IEEE, 1985, pp. 383-395.

14. Cohen, J.C. (Benaloh) and Fischer M.J. A Robust and Verifiable Cryptographically Secure Election Scheme. Proc. 26th FOCS, IEEE, 1985, pp. 372-383.

15. DeMillo, R.A., N. Lynch and M. Merritt. Cryptographic Protocols. 14th STOC, ACM-SIGACT, May, 1982, pp. 383-400.

16. Diffie, W., and Hellman M.E. "New Directions in Cryptography". *IEEE Transactions of Information Theory IT-22* (November 1976), 644-654.

17. Even, S., Goldreich O. and Lempel A. "A Randomized Protocol for Signing Contracts". *Communications of the ACM 28*, 6 (June 1985), 637-647.

18. Feige, U., A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity. 19th STOC, 1986, pp. 210-217.

19. Fischer, M., S. Micali, C. Rackoff, and D. Wittenberg. An Oblivious Transfer Protocol Equivalent to Factoring. Manuscript, 1986.

20. Galil, Z., Haber S. and Yung M. A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems. Proc. 26th FOCS, IEEE, 1985, pp. 360-371.

21. Goldreich, O., S. Goldwasser, and S. Micali. How to Construct Random Functions. Proc. 25th FOCS, IEEE, 1984, pp. 464-479.

22. Goldreich O., and R. Vainish. How to Solve Any Protocol Problem: an Efficiency Improvement. These proceedings.

23. Goldreich, O., S. Micali and A. Wigderson. Proofs that Yield Nothing But their Validity and a Methodology of Cryptogrphic Protocol Design. 27th FOCS, IEEE, October, 1986, pp. 174-187.

24. Goldreich, O., S. Micali and A. Wigderson. How to Play Any Mental Game. 19th STOC, 1987, pp. 218-229.

25. Goldwasser, S. and Micali S. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. Proceedings of the 14th Annual ACM Symp. on Theory of Computing, ACM-SIGACT, May, 1982, pp. 365-377.

26. Goldwasser, S., S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. 17 STOC, ACM-SIGACT, May, 1985, pp. 291-304.

27. Halpern, J. and Rabin M.O. A Logic to Reason about Likehood. Proc. 15th STOC, ACM, 1983, pp. 310-319.

28. Hastad, J. and A. Shamir. The Cryptographic Security of Truncated Linearly Related Variables. 17th STOC, ACM-SIGACT, May, 1985, pp. 356-362.

29. Impagliazzo R., and M. Yung. Direct Minimum-Knowledge Computations. These proceedings.

30. Kranakis, E.. *Primality and Cryptography*. John Wiley and sons, Chichester, 1986.

31. Levin, L. One-way Functions and Pseudorandom Generators. Proc. 17th STOC, ACM, 1985.

32. Luby, M. , Micali S. and Rackoff C. How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin. 24 FOCS, IEEE, November, 1983, pp. 11-22.

33. Pippenger, N., and M.J. Fischer. "Relations among Complexity Measures". *Journal of the ACM 26* (1979), 361-381.

34. Rabin. M. O. Digitalized Signatures and Public-key Functions as Intractable as Factorization. LCS/TR-212, MIT, January", 1979.

35. Rivest, R. , Shamir A., Adleman L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems". *Communications of the ACM 21*, 2 (February 1978), 120-126.

36. Shamir, A. "How to Share a Secret". *Communications of the ACM 22*, 11 (November 1979), 612-613.

37. Shamir, A., Rivest R. Adleman L. Mental Poker. In *Mathematical Gardner*, Klarner D. E., Ed., Wadsworth Intrntl, 1981, pp. 37-43.

38. Vazirani, U. and Vazirani V. Efficient and Secure Pseudo-Random Number Generation. Proc. 25th FOCS, IEEE, 1984, pp. 458-463.

39. Vazirani, U. and Vazirani V. Trapdoor Pseudo-random Number Generators, with Applications to Protocol Design. 24th FOCS, IEEE, November, 1983, pp. 23-30.

40. Yao, A. Protocols for Secure Computations. 23rd FOCS, IEEE, November, 1982, pp. 160-164.

41. Yao, A. Theory and Applications of Trapdoor Functions. 23rd FOCS, IEEE, November, 1982, pp. 80-91.

42. Yao, A. How to Generate and Exchange Secrets. 27th FOCS, IEEE, October, 1986, pp. 162-167.

GRADUAL AND VERIFIABLE RELEASE OF A SECRET

(Extended Abstract)

Ernest F. Brickell

Bell Communications Research;
435 South Street; Morristown, NJ 07960; USA

David Chaum
Ivan B. Damgård
Jeroen van de Graaf

Centre for Mathematics and Computer Science
Kruislaan 413; 1098 SJ Amsterdam; The Netherlands.

Abstract:
Protocols are presented allowing someone with a secret discrete logarithm to release it, bit by bit, such that anyone can verify each bit's correctness as they receive it. This new notion of *release of secrets* generalizes and extends that of the already known *exchange of secrets* protocols. Consequently, the protocols presented allow exchange of secret discrete logs between any number of parties.

The basic protocol solves an even more general problem than that of releasing a discrete log. Given any instance of a discrete log problem in a group with public group operation, the party who knows the solution can make public some interval I and convince anyone that the solution belongs to I, while releasing no additional information, such as any hint as to where in I the solution is.

This can be used directly to release a discrete log, or to *transfer* it securely between different groups, i.e. prove that two instances are related such that knowledge of the solution to one implies knowledge of the solution to the other.

We show how this last application can be used to implement a more efficient release protocol by transferring the given discrete log instance to a group with special properties. In this scenario, each bit of the secret can be verified by a single modular squaring, and unlike the direct use of the basic protocol, no interactive proofs are needed after the basic setup has been done.

Finally, it is shown how the basic protocol can be used to release the factorization of a public composite number.

1. Related work

Two of the first contributions to the field of *secrets exchange* were made by Blum[BL81] and Rabin[Ra81]. They both showed how to exchange secrets using oblivious transfer. Later, Blum[Bl83] proposed his well known protocol for exchange of factorizations; see Hastad and Shamir [HaSh85] for a discussion of some problems with this protocol. It cleverly expoits the fact that the factorization of an integer is a secret of many bits. In the original version, one participant would sometimes know one secret bit more than the other. Tedrick's work [Te83], [Te84] is one approach to reducing the amount of unfairness introduced by this fact. A different approach to exchanging one bit was taken by Luby, Micali and Rackoff [LMR83] and Vazirani and Vazirani [VaVa83] Here, the information exchanged is probabilistic, i.e. in each round of the protocol additional certainty is gained about the value of a single secret bit. The work of Yao [Ya86] implies the existence of extremely general two party exchange-of-secret protocols, in connection with the problem of secure computations with private inputs.

While the practical proposals in the literature involve disclosure of secret factorizations (combined with square roots), the present protocol allows disclosure of discrete logarithms. This is of practical importance for protocols based on secret discrete logs, since means to *transfer* a secret from a discrete log form to a factorization form are not known.

Perhaps a more fundamental difference between the related literature and the present protocol, however, is that they only provide for *exchange* of secrets, while we allow the more general *release* of secrets (one possible exception is [VaVa83]. Still, the fact that the information released by our protocol is deterministic, remains a fundamental difference between the protocols). In earlier exchange of secrets protocols, it was either inherent in the protocol that an exchange had to take place [LMR83], or the work done by the party with the secret would increase rapidly with the number of parties who were to recieve it [Bl83]. In a release protocol the party with the secret is essentially just broadcasting information about it. Therefore, a release can be simultaneous to any number of parties, and can be readily used to implement an exchange of secrets between any number of parties.

The theoretical existence of of such release-of-secrets protocols can be proved using techniques from the work of Brassard and Crepeau [BrCr86], of Chaum [Ch86] and of Goldreich, Micali and Wigderson [GMW86]. But since their methods (as Yao's [Ya86]) involve individual processing of all gates in a boolean circuit or a reduction to 3COL, the resulting protocols would be impractical. We propose a more practical solution to the problem: in the most efficient version, to release one bit, one modular square root has to be computed, while the correctness of that bit can be checked by one squaring.

2. Showing membership of an interval.

Suppose one party — the prover (P) — knows a solution z to the equation $\alpha^z = \beta$ where α and β belong to some group with a public group operation. One obvious example of this is the multiplicative subgroup of the integers modulo p, denoted as Z_p^*, where p is a large prime. Suppose also that $z \in [a, a+B]$, where a and B are public. The protocol below can then be used by the prover to convince someone else — the verifier (V) — that $y \in [a-B, a+2B]$.

Although we talk about a single verifier for convenience, it should be noted that any set of participants could easily play the part of V without loosing the efficiency of the protocol. This is so because the role of V is passive: he flips coins (in public) and expects answers from P according to the coinflips. Thus, this protocol has the basic properties we require from a release of secrets protocol.

An important tool used in the protocol is a *bit commitment scheme.* When a party P commits to a bit, this means that he encrypts a 0 or 1 in such a way, that

— the other party cannot tell what the value of the bit is;

— P cannot later change his mind about the bit.

— P can convince anyone about the value of the bit. This is referred to as *opening* the commitment.

For the purpose of the protocol presented in this paper, it is not necessary to fix a particular choice of bit commitment scheme. The above properties will be sufficient to see why the protocol below is minimum knowledge. In [CDG87] bit commitment schemes are discussed in more detail; also a specific example is given that relies on the hardness of the discrete log problem. More precisely, with this scheme a commitment releases no information in the Shannon sense about the bit it contains, and the party committing to a bit cannot change it, unless he can solve a discrete log problem *before the protocol is over.* With this scheme, the protocol below will be *perfect zero knowledge* in the terminology of [GMW86] i.e. the secret of the prover is unconditionally protected, but the verifier will only be convinced modulo his belief that the prover has not been able to solve the hard problem that prevents him from cheating. Note that this flavour of commitment scheme does not fit into the original model of [GMR85], where the prover has infinite computing power. The term "perfect zero knowledge" therfore only referres to the fact that a simulation will produce the exact same probability distribution as the actual protocol. Other choices of commitment schemes (like one based only on the existence of a one-way permutation) will imply that the verifier will always be convinced with an exponentially small residue of doubt, and that the protocol will be "ordinary" zero knowledge, even with an infinitely powerful prover. In this case, however, the secret of the prover will only be conditionally protected, and will be revealed if the cryptographic assumption is broken at any point after completion of the protocol. For a more detailed discussion of these problems, consult [BCC87].

Having chosen a bit commitment scheme, the prover can commit to a string of bits by simply computing commitments bit by bit. For the bitstring s, the resulting string of commitments is denoted $BC(s)$.

PROTOCOL RELEASE INTERVAL

1. P picks $t_1 \in [0,B]$ and computes $t_2 = t_1 - B$.
   P computes a bit commitment of α^{t_1} and of α^{t_2}, and sends the unordered pair $(BC(\alpha^{t_1}), BC(\alpha^{t_2}))$ to V.

2. V flips a coin and requests either
   (a) to receive t_1 and t_2 and opening by P of both $BC(\alpha^{t_1})$ and $BC(\alpha^{t_2})$, or
   (b) to receive $(z+t_i)$ for $i = 1$ or 2, and opening by P of $BC(\alpha^{t_i})$

3. On request (a) P sends t_1 and t_2, and opens both commitments.
   On request (b) P sends either $(z+t_1)$ or $(z+t_2)$, whichever is in the interval $[a, a+B]$, and opens the corresponding bit commitment.

4. On request (a), V checks that the two commitments did indeed contain α^{t_1} and α^{t_2}, that $t_1 \in [0,B]$, and that $t_1 - t_2 = B$.
   On request (b), V checks that α^{z+t_i} equals $\beta\alpha^{t_i}$ (he knows α^{t_i} as a result of the opening of one of the commitments).

Steps 1-4 are repeated n times.

Lemma 1: If the prover cannot change the contents of his commitments after step 1, and is able to satisfy both request (a) and request (b), then $z \in [a-B,\ a+2B]$.

Proof: The result follows immediately from $t_1 \in [0,B]$, $t_2 \in [-B,0]$, and $z+t_i \in [a,\ a+B]$ for either $i=1$ or 2. □

By this lemma, if the protocol is repeated n times, then the verifier will be convinced with confidence $1-2^{-n}$, assuming that the prover could not change the content of his bit commitments. As mentioned before, changing the content of a commitment is either impossible or occurs with only negligible probability, depending on the choice of bit commitment scheme.

Note that there is a discrepancy between the requirement for z, namely $z \in [a, a+B]$, and what is actually proven, namely $z \in [a-B, a+2B]$. Below we will show that only if $z \in [a, a+B]$, no knowledge is released by the above protocol.

There are several related definitions for minimal knowledge, which are quite technical. In the full paper, we will give all of the formal definitions and the proof that our protocol is minimum knowledge, but for this extended abstract, we will only give intuitive definitions and proofs.

The concept of zero knowledge proofs was introduced by [GMR85]. Rougly speaking, they defined *knowledge* to be something that cannot be computed in random polynomial time. Essentially, they call a proof that a string x is in a language L, a *zero knowledge proof* if a probabilistic polynomial time turing machine will be convinced (with high probability) that $x \in L$, but will learn nothing other than this one bit of knowledge.

To describe what we know about the discrete logarithm release protocol, we need to change the [GMR85] definition a little: First we will, as in [GHY87] and [FFS87], assume that both the prover and the verifier are restricted to polynomial time computations. This corresponds better to a practical situation, and allows use of both types of bit commitment schemes mentioned above. Secondly, we will say that a protocol is a (L_1,L_2) minimum knowledge protocol if the verifier will be convinced (with high probability) that $x \in L_1$ but the verifier will receive no knowledge other than the fact that $x \in L_2$. More precisely, the definition is satisfied if it is possible, assuming only that $x \in L_2$, to simulate a protocol conversation such that no probabilistic polynomial time algorithm can tell the difference between a simulated and a genuine conversation. This definition allows for a discrepancy between what the verifier is told and what he will be convinced of. It reduces to the [GMR] definition of zero knowledge when $L_1 = L_2$, apart from the assumption on the computational power of the prover.

We are now ready to give our informal proof that the discrete logarithm release protocol is an (L_1,L_2) minimum knowledge protocol, where $L_1 = \{\alpha^x \mid x \in [a-B,a+2B]\}$ and $L_2 = \{\alpha^x \mid x \in [a,a+B]\}$.

Lemma 2: If $z \in [a,a+B]$, then the distribution of the value of $(z + t_i)$ sent in step 3(b) of RELEASE INTERVAL is independent of the value of z.

Proof: It is trivial to check, that if some number $d \in [a,a+B]$ is sent as $(z + t_i)$, then exactly one value in $[0,B]$ must have been chosen as t_1. So if t_1 is chosen uniformly in $[0,B]$, then $(z + t_i)$ must be uniform in $[a,a+B]$ □.

Lemma 3: There exists a simulator for RELEASE INTERVAL.

Proof: Using the following observation it is easy to see that the protocol can be simulated. Since the protocol proceeds in rounds, the simulator (P') can rewind (i.e. stop and go back to a previous state) the protocol if the simulation gets stuck at any point. Thus, we can essentially assume that the simulator knows in advance which choice the verifier will make in step 2. It is therefore trivial to compute a message for step 1 that will satisfy the verifier if he is going to make choice (a). For choice (b) the simulator chooses $d \in [a,a+B]$ at random, to serve as $z+t_i$. It then puts $\gamma = \alpha^d\beta^{-1}$. Now P' can commit to γ, and to something totally random that has the same number of bits as γ. These commitments are sent in step 1; in step 3, d is sent in place of $z + t_i$, and the bit commitment containing γ is opened.

If unconditionally secure bit commitments are used, it is now clear from Lemma 2 that the simulated conversation will have exactly the same distribution as the actual protocol conversation.

If the other type of commitments mentioned above is used, i.e. a scheme based on probabilistic encryption, the distributions will be different, but the difference cannot be recognised in polynomial time, if the encryption function is secure against polynomial time attacks. □

P is of course free to choose $z \in [a+B, a+2B]$, but then the protocol releases information. For instance, if z is almost equal to $a+2B$, P must always choose t_1 close to 0, and release $(z+t_2)$ which is close to $a+B$. We could, however, take away P's freedom to choose t_1 himself. The protocol could start by a sequence of coinflips with the property that both parties trust their randomness, but only P gets to see the outcome. This can easily be implemented using bit commitments. The protocol would now proceed as before, but in step 3(a), P would have to prove that he computed t_1 from the coinflips generated initially. This modification means that there is now a non-zero probability of catching P if z is not in $[a, a+B]$. This probability is larger, the further away z is from $[a, a+B]$, and will tend to 0 with increasing n for any fixed z not in $[a, a+B]$. In fact, this means that we can get exponentially large confidence about any interval that properly contains $[a, a+B]$, at the cost of a larger n. More details will be given in a full version of this paper.

2.1 Release Interval Without Bit Commitments.

There is a variation of RELEASE INTERVAL that does not require the use of bit committment. This variation makes use of a parameter δ which we will assume is equal to ϵB for some $\epsilon \in (0,1)$. The protocol proceeds as follows:

RELEASE INTERVAL II

The prover

1. Picks $t \in [B, B+\delta]$
2. picks $t_1 \in [0, t]$ and computes $t_2 = t_1 - t$.
3. permutes t_1 and t_2.
4. Sends the unordered pair α^{t_1} and α^{t_2} to the verifier.

The verifier

5. Requests either (a) to receive t_1 and t_2 or (b) to receive $z+t_i$ for $i = 1$ or 2.

The prover

6. On request (a) sends t_1 and t_2. On request (b) sends either $z+t_1$ or $z+t_2$, whichever is in the interval $[a, a+t]$.

The verifier checks

7. on request (a), that the prover did indeed send α^{t_1} and α^{t_2}, and that $|t_1 = t_2| \in [B, B+\delta]$.
8. on request (b), that α^{z+t_i} equals either $\beta\alpha^{t_1}$ or $\beta\alpha^{t_2}$, and that $z+t_i \in [a, a+B+\delta]$.

After RELEASE INTERVAL II has been repeated n times, the verifier will be convinced with confidence $1-\frac{1}{2^n}$ than $z \in [a-B-\delta, a+2B+2\delta]$, but this interval can be shortened if the verifier trusts the randomness of t and t_1. RELEASE INTERVAL II is more efficient than RELEASE INTERVAL. Further, assuming that

— the verifier does not have a probabilistic polynomial time algorithm that solves discrete log given that the solution is in an interval of size B

— $z \in [a, a+B]$,

then RELEASE INTERVAL II is minimum knowledge (but clearly not perfect zero knowledge). In other words, if the verifier cannot find the secret directly from the public information, then he is in essentially the same situation after having executed the protocol. The proof of this is quite long and technical and has therefore been omitted in this extended abstract.

3. Two applications

It is easy to see, how the RELEASE INTERVAL protocol can be used to release information about the solution to a discrete log problem in a verifiable way, starting with the high order bits: Use protocol RELEASE INTERVAL many times with ever decreasing values of B. This convinces the verifier that the solution is contained in smaller and smaller intervals.

If it is easy to compute square roots in the group involved, we can also release the solution, starting with the low order bits. Consider for instance Z_p^*, with p prime. Fix some generator α of Z_p^*. For any square $\beta \in Z_p^*$, we let the principal square root of β be the root with discrete log base α smaller than $\frac{p-1}{2}$. It is easy to see that the discrete log of any number can be computed bit by bit, starting with the low order bits, given an oracle that decides which of two square roots is the principal one. At each point in this computation, someone who already knows the discrete log could serve as the oracle by using RELEASE INTERVAL to prove the validity of his answers. This clearly leads to a protocol that releases the discrete log starting with the low order bits.

4. An Efficiency Improvement.

In the previous two applications we had to go the RELEASE INTERVAL once for each bit released. This can be quite time consuming in general because each instance of RELEASE INTERVAL involves a cut-and-choose with many iterations. Using a technique to transfer a discrete log from one group to another, we can reduce this to 1 application of RELEASE INTERVAL. However, besides the protocol needed to transfer the logarithm, we the need other protocols to prove that the groups used have specific properties. Recall that the secret z is determined by $\alpha^z = \beta$, where $\alpha, \beta \in Z_p^*$. Basically we transfer the secret from $\langle\alpha\rangle$ to $\langle\alpha_1\rangle$, where $\alpha_1 \in Z_N^*$ and N is a composite. α_1 is chosen, such that 2^l divides $order(\alpha_2)$, where $l = {}^2\log p$. Then the secret is released just by showing square roots in Z_N^*, which are easily verified but hard to compute without the factorization. First we will show how P can prove to V that N has the required properties. This is done in steps 1-3 below. Note that these steps are only necessary once, i.e. if many secrets are to be released, the same N and α_1 can be used for all of them.

Step 1:
Party P chooses a modulus $N = qr$, where q and r are large primes, constructed such that P knows the factorization of $q-1$ and $r-1$, and such that 2^l divides at least one of $q-1$ and $r-1$. This can easily be done , e.g. by a variation of Gordon's method [Go84].

Step 2:
Let QR denote the set of quadratic residues modulo N. Next P selects a random element $\alpha_1 \in Z_N^*$ with $J(\frac{\alpha_1}{N}) = 1$ and α not in QR. From the Chinese Remainder Theorem it can easily be seen that such an α_1 will always have the maximum possible 2-power dividing its order, so 2^l is certainly a divisor of the order of α_1.

Step 3:
P makes public N and α_1. Then

a) P proves that α_1 is a quadratic non-residue, e.g. by using the protocol in [GMR85].

b) P proves that 2^l divides $order(\alpha_1)$:

i) V choses a random odd number r, with $0 < r < R$, where R is fixed (the choice of R is considered later). V chooses $b \in \{0,1\}$.

ii) V sends to P:
$\alpha_1^{r \cdot 2^k}$ if $b = 1$;
$\alpha_1^{r \cdot 2^{k-1}}$ if $b = 0$.

iii) Since P has constructed N, he can compute the order of what he receives. Based on the outcome he determines b and sends it to V.

iv) V checks that the correct value of b was returned.

Steps i through iv are repeated n times.

It is easy to see that if P is not cheating, he can always give the correct answer in step iii) Consider the situation where P is trying to cheat, i.e. the maximum 2-power dividing $order(\alpha_1)$ is 2^c, where $c < l$. Let G be the maximal sub-group of $\langle\alpha_1\rangle$ of odd order. Clearly, $G = \langle\alpha_1^{2^k}\rangle = \langle\alpha_1^{2^{k-1}}\rangle$, which means that both $\alpha_1^{r \cdot 2^k}$ and $\alpha_1^{r \cdot 2^{k-1}}$ will look to P like randomly chosen elements of G. Since V does not know the order of G, he cannot make the distribution completely uniform over G, but he can make as close an approximation as he likes by choosing the R from step i) large enough. Assuming that P cannot distinguish the distribution of $\alpha_1^{r \cdot 2^k}$ and $\alpha_1^{r \cdot 2^{k-1}}$, he can do no better than guesing at random in step iii) whence he will be caught with probability $1-2^{-n}$.

From V's point of view the protocol is not zero knowledge as it stands. V could use it to get some — probably useless — information about other elements in Z_N^* than α_1. The protocol can be modified such that it is zero-knowledge. This involves ensuring that V knows the discrete log with respect to α_1 of the numbers he send to P. This could be accomplished using for example the general discrete log protocol described in [CEGP86]. We also have other methods which will work well in this special case, but we omit the details here for simplicity.

Step 4:
P transfers the problem from Z_p^* to Z_N^*. Remember P's secret is the solution z of $\alpha^z = \beta_1$ in Z_p^*. Now P computes $\beta_1 = \alpha_1^z$ in Z_N^* and makes β_1 public. Using RELEASE INTERVAL in the cross-product of $\langle\alpha\rangle$ and $\langle\alpha_1\rangle$, P can prove that he knows a $z \in [1, p-1]$, which solves both $\alpha^z = \beta$ and $\alpha_1^z = \beta_1$.

Step 5:
To release a single bit (the least significant bit of z), P proceeds as follows:

if $z = 2z'$, he makes public a square root of β_1.

if $z = 2z'+1$, he makes public a square root of $\beta_1 \alpha_1^{-1}$.

Replace z by z', and β_1 by the square root released.

This step works because of the following two easily verified facts:

1. The l least significant bits of z are uniquely determined by the equation $\alpha_1^z = \beta_1$.
2. If α_1 is a non square mod N and $\alpha_1^z = \beta_1$, then z is even if and only if β_1 is a square mod N.

By fact 1, the pair (α_1, β_1) determines exactly one secret of size l bits, and by fact 2, anyone can check on P by squaring the numbers released.

Regarding the security of this protocol, notice first that steps 2 and 3 can be executed in zero knowledge. Moreover, the release of α_1 cannot endanger the factorization of N, since anyone can select at random a number of Jacobi symbol 1 mod N, which with probability ½ will have the same properties as α_1. It is clear that breaking this scheme can be no harder than factoring numbers of the required special form: $N = qr$, where 2^l divides at least one of $q-1$ and $r-1$. It is not known, however, whether these numbers are easier to factor than randomly chosen RSA-moduli. But with respect to the factoring algorithms known at present, the factorization of N should be safe, if $q-1$ and $r-1$ contain large prime factors, in addition to the 2-powers. Finally, it is also conceivable that someone could try to attack the equations $\alpha^z = \beta$ and $\alpha_1^z = \beta_1$ directly, without factoring N. But this involves solving a discrete log problem, and we know of no argument indicating that discrete log should be easier in this special case than in the general situation.

6. Releasing a factorization.

This final application allows a prover to release the factorization of a public modulus N. It is wellknown that knowledge of a multiple of $\phi(N)$ suffices to factor N easily. Using this fact, the protocol proceeds as follows: The verifier chooses an integer T and a set of elements $a_1, \ldots, a_s$ in Z_N^*. The prover then computes $t = T \bmod \phi(N)$, and sends $b_i = a_i^t$ to the verifier, who can check these numbers by raising the a's to the T'th power. The prover then uses the RELEASE INTERVAL protocol to convince the verifier that he knows a simultaneous solution to all s discrete log instances which belongs to an interval NOT including T. When the prover uses RELEASE INTERVAL more times to release bits of t, the verifier can be convinced, that he will get at least a multiple of $LCM(\text{order }(a_1),..,\text{order }(a_s))$. The probability that this number equals the exponent of the group Z_N^* (i.e. the smallest integer e such that $g^e = 1$ for all elements in the group) is exponentially large in s, and it is not hard to see that *exponent*(Z_N^*) will serve as well in factoring N as $\phi(N)$.

Acknowledgement.

The first author would like to thank Oded Goldreich for discussions related to this work.

References

[Bl81] Blum: "Three applications of the oblivious transfer", Dept. of EECS, Univ. of California, Berkely, 1981.

[Bl83] Blum: "How to exchange (secret) keys", *ACM Transactions on Computer Systems*, vol. 1, 1983, pp. 175-193.

[BCC87] Brassard, Chaum and Crépeau:"Minimum disclosure proofs of knowledge", to appear.

[BrCr86] G. Brassard, and C. Crépeau, "Zero-Knowledge Simulation of Boolean Circuits," *Presented at Crypto 86*, (August 1986).

[Ch86] D. Chaum, "Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How," *Presented at Crypto 86*, (August 1986).

[CDG87] Chaum, Damgård and van de Graaf: "Multiparty computations ensuring privacy of each party's input and correctness of the result", Proc. of Crypto 87.

[CEGP86] D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta, "Demonstrating possession of a discrete logarithm without revealing it," *To appear in the Proceedings of Crypto 86*, (August 1986).

[CG87] Chaum, van de Graaf: "An improved protocol for demonstrating possession of a discrete log and some generalisations", Proc. of Eurocrypt 87.

[FFS87] Fiege, Fiat and Shamir: "Zero knowledge proof of identity", Proc. of STOC 87.

[GHY87] Galil, Haber and Yung: "Cryptographic Computation: Secure Fault-tolerant Protocols in the Public Key Model", Proc. of Crypto 87.

[GMR85] S. Goldwasser, S.Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *17th STOC* (1985).

[GMW86] O. Goldreich, S. Micali, and A. Wigderson, "How to Prove all NP-statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design," *Presented at Crypto 86*, (August 1986).

[Go84] J. Gordon, "Strong primes are easy to find", Proceedings of Eurocrypt 84.

[HaSh85] "The cryptographic security of truncated linearly related variables," *Proc. of 17th STOC*, 1985, pp. 356-362.

[LMR83] Luby, Micali and Rackoff: "How to simultaneously exchange a secret bit by flipping a symmetrically biased coin", Proc. 24th FOCS, 1983, pp.11-21.

[Ra81] Rabin: "How to exchange secrets using oblivious transfer", Technical memo, TR-81, Aiken Computation Lab., Harward Univ., 1981.

[VaVa83] Vazirani and Vazirani: "Trapdoor pseudo random number generators with applications to protocol design", Proc. 24th. FOCS, 1983, pp.23-30.

[Te83] Tedrick: "How to exchange half a bit", Proc. of Crypto 83, pp.147-151.

[Te84] Tedrick: "Fair exchange of secrets", Proc. of Crypto 84, pp.434-438.

Strong Practical Protocols[†]

Judy H. Moore
Sandia National Laboratories
Albuquerque, NM 87185

Recent progress in the area of cryptography has given rise to strong cryptoalgorithms using complex mathematical systems. These algorithms often require quite sophisticated computing capabilities for their implementation and are designed to withstand attack by equally sophisticated opponents with nearly unlimited resources available to them. However, the mere existence of such algorithms is not enough to solve the problems of message secrecy and authentication. The procedures for handling the data, including the use of a cryptoalgorithm, must insure that the desired level of security is achieved. Such a set of rules or procedures is known as a cryptographic *protocol*.

There have been many examples in the literature ([2],[3],[6],[7],[10], [11]) in the past few years of protocol failures. These examples sometimes involve the cryptanalysis of the particular instance of the cryptosystem used, but often merely demonstrate the failure of the protocol to provide the advertised level of security and/or authentication.

Research in the area of cryptographic protocols has been considering methods to provide "provably secure" protocols ([1],[4],[5],[8],[9]). Progress has been made, although, as might be expected, the initial results either apply to systems which have been idealized in order to simplify the problem or use techniques to gain security which are not practical for application. These techniques are obtained using abstract principles designed to model ideal security systems. However, when cryptosystems are actually fielded, much less abstract principles are applied. Rather than trying to model ideal systems to provide "provably secure" protocols, we propose the development of techniques for use by protocol designers to precisely define the security provided by the protocol.

Modern systems designed to protect information can make use of cryptography, physical barriers and procedural constraints to accomplish their goals. It is important to note that, in general, elements from each of these three areas are required for optimal

†This work performed at Sandia National Laboratories supported by the U. S. Department of Energy under contract no. DE-AC04-76DP00789.

protection. Perhaps because the physical barriers seem more concrete, the analysis of their effectiveness appears to be easier than the analysis of the cryptographic portions of the protocol. Such an analysis giveS rise to some broad guidelines for the purpose behind any element in a protection system. In particular, it seems that the value of any security measure lies in its capacity to contain the threat within a well-defined and reasonable boundary. The guardian of an object utilizes a security measure, not because it will provide absolute security, but rather because it can limit the threat and clearly define the boundary of the security problem remaining. For example, the purpose of a barbed wire fence around a secure area is to limit the attention of the guards to that well-defined boundary. It also limits the threat to those willing and able to confront the barbed wire. The guard force is still necessary since the fence cannot provide absolute security. Similarly, the use of a secure encryption algorithm to maintain secrecy in communications between two parties, does not eliminate the need for some information to be kept secret. At best, it transfers the need for protection to the key, which should be smaller and more easily controlled than the messages themselves.

From the above discussion, we abstract the following principle for application to the development and analysis of protocols. Each component of a protocol should serve to either limit the number of opponents capable of posing a threat or limit the exposure to these opponents of the protected data. The protocol designer must attempt to demonstrate how the protocol accomplishes this and to what degree it can be expected to provide the desired security. It is important to notice that the key word used here is **limit**, not **eliminate**. This reflects the effect of what is probably an axiom; namely, that **we cannot eliminate the problem; we can only bound the problem**. Absolute security, provable security may not be required, but careful analysis of the how the critical components bound the problem is essential in order to assure that, when the protocol is actually fielded, the declared level of security is not degraded.

To illustrate this kind of analysis, consider the example of a communications network using a public key cryptosystem. We assume a Central Keying Authority (CKA) which is responsible for issuing to each subscriber in the network, a cryptosystem designed to provide a secrecy channel for communications with that subscriber. The CKA also publishes the public parameters of each subscriber's cryptosystem. We will use the notation $E_i(M)$ to denote the encryption of a message M using the published parameters of the ith subscriber. The decryption of a ciphertext C using the secret parameters will be denoted by $D_i(C)$. In order to send a message M to subscriber i, a transmitter looks up the public encryption parameters and sends $E_i(M)$ across open communication lines.

The specification of the public key algorithm to be used is of course critical her but much can be said before proceeding further. The critical components to be analyzed this level of specification are:

1. The CKA must choose good parameters to insure that the best possible security achievable with the chosen cryptosystem. Without this requirement, the cryptosystem cannot serve to limit the threat. The CKA must also be trusted not to divulge each subscriber's secret parameters since failure to accomplish this may result in exposure t the threat of the data to be protected.

2. The CKA must publish and maintain the integrity of the public parameters. Thi is necessary not only to facilitate the communications network, but also to limit the exposure to the threat of the data, since the insertion of fraudulent information in th directory would enable an unauthorized user to pose as an authorized subscriber.

3. The subscribers must take appropriate measures to maintain the secrecy of thei decryption functions in order to limit the threat by requiring some effort to decrypt t messages.

Since the purpose of the described protocol is to allow for messages which are not understandable by an outsider to be sent to a given subscriber, the remaining analysis required involves the description of the extent to which this is possible. For example, if $E_i(M)$ is sent to subscriber i, the meaning of M might be exposed if:

1. E_i is not a strong cryptofunction, so that a decryption of $E_i(M)$ is possible by a simple cryptanalysis of the algorithm. For example, if RSA is used, the threat is potentially limited to those capable of factoring the modulus. If the parameters are properly chosen, this threat can be made quite small. However, the protocol designer and/or analyser must be careful to avoid the pitfall of claiming that **finding the meanin of M means breaking the cryptosystem** E_i, as the remaining points will address.

2. $E_i(M)$ may be jibberish to the outsider, but the meaning of M may still be discernable. This, of course, is possible if the number of messages that can be meaningful to the subscribers is small so that precalculation of $E_i(N)$ for all possibl N could make the meaning clear by a simple table lookup [11]. Therefore, the message space must be large enough to limit this threat to the desired level.

3. $E_i(M)$ might be used together with a collection $\{E_j(M)\}$ of encryptions of the same message to other subscribers. In the case of an RSA system, this is possible when a common modulus for each subscriber or if a small common encryption exponent is used [3],[6],[10].

4. $E_i(M)$ might be used together with $\{E_j(M_j)\}$, where M_j is some known variant of M, to reconstruct M. For example, if M_j is M together with a time stamp and the encryption function is RSA with a small exponent [6].

Each of these represents a known problem for some or all public key cryptosystems which are also logically obvious areas of concern. The protocol designer should therefore take care to consider each of these in the specification of the cryptoalgorithm to be used and the parameters to be chosen.

It is important for the protocol designer to be precise in the statement of the security provided by each component of the protocol. In practical application, broad assumptions need to be replaced by specific measures of the level of degradion of security if the the given condition is not met. For a simple illustrative example, the statement that the CKA must be trusted to maintain the secrecy of the subscribers' decryption functions, should really be replaced by the statement that no messages for a given subscriber can be secret if the CKA has divulged the decryption parameters of the subscriber. While this is of course obvious, if the protocol designer adopts this style of describing the protocol, the security issues become clearly demarked.

When protocols are designed, some assumptions about the setting or the cryptoalgorithm used are inevitably made. There is no problem for a practical protocol application with such assumptions, as long as these are clearly defined. In fact, the failure to identify some assumptions is quite often the root of protocol failures, so that a clear definition of all assumptions is necessary for the development of strong protocols. However, one further step would be very useful to both the designer and the applier of a protocol. This step would entail the analysis and possible quantification of the effect on the protocol if some identified assumption is not valid. The benefits of this approach are twofold. First, when the protocol is applied, the advertised level of security will not be degraded by the failure to meet some unspecified requirements for the system. Secondly, the protocol designer will have a more clearly defined protocol and may find that unnecessary assumptions could be discarded, making the protocol more widely applicable.

By way of summary, what we actually propose is a change in perspective in two areas for the design and analysis of protocols. First, for each element of the protocol, the following questions should be answered:

1. How does this element limit the threat or the exposure to the threat?
2. To what extent does this element actually meet that goal?

Secondly, when assumptions are made, they must be clearly defined and the level of security lost when the setting for an application of the protocol does not satisfy the assumption should also be clearly defined.

References

1. R. Berger, S. Kannan and R. Peralta, 'A Framework for the Study of Cryptographic Protocols', *Advances in Cryptology - Proceedings of Crypto 85*, pp.87-103.

2. G.I. Davida, 'Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem', Tech. Rep. TR-82-2, Dept. of Electrical Engineering and Computer Science, Univ. Of Wisconsin, Milwaukee, WI, Oct. 1982.

3. J.M. DeLaurentis, 'A Further Weakness in the Common Modulus Protocol for the RSA Cryptoalgorithm' *Cryptologia*, Vol. 8, No. 3, July 1984, pp. 253-259.

4. S. Even, O. Goldreich and A. Shamir, 'On the Security of Ping-Pong Protocols Using the RSA', *Advances in Cryptology - Proceedings of Crypto 85*, pp.58-72.

5. S. Goldwasser, S. Micali and A. Yao, 'Strong Signature Schemes', *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 1983, pp.431-439.

6. J. Hastad, 'On Using RSA with Low Exponent in a Public Key Network', *Advances in Cryptology - Proceedings of Crypto 85*, pp.403-408.

7. R.R. Jueneman, S.M. Matyas and C.H. Meyer, 'Message Authentication with Manipulatior Detection Codes', *Proceedings of the 1983 Symposium on Security and Privacy*, pp.33-54.

8. M. Merritt, *Cryptographic Protocols*, Ph. D. Thesis, Georgia Institute of Technology, GIT-ICS-83/06,1983.

9. J.K. Millen, S.C. Clark and S.B. Freedman, 'The Interrogator: Protocol Security Analysis', *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2, February 1987, pp.274-288.

10. G.J. Simmons, 'A "Weak" Privacy Protocol Using the RSA Cryptoalgorithm', *Cryptologia*, Vol. 7, pp. 180-182.

11. G.J. Simmons and D.B. Holdridge, 'Forward Search as a Cryptanalytic Tool Against a Public Key Privacy Channel', *Proceedings of the IEEE Computer Society 1982 Symposium on Security and Privacy*, 1982, pp.117-128.

Identity-based conference key distribution systems

Kenji Koyama Kazuo Ohta*

NTT Basic Research Laboratories
Nippon Telegraph and Telephone Corporation
3-9-11, Midori-cho, Musashino-shi, Tokyo, 180 Japan

*NTT Communications and Information Processing Laboratories
Nippon Telegraph and Telephone Corporation
1-2356, Take, Yokosuka-shi, Kanagawa, 238-03 Japan

Abstract : This paper proposes identity-based key distribution systems for generating a common secret conference key for two or more users. Users are connected in a ring, a complete graph, or a star network. Messages among users are authenticated using each user's identification information. The security of the proposed systems is based on the difficulty of both factoring large numbers and computing discrete logarithms over large finite fields.

1. Introduction

Shamir first proposed the idea of identity-based cryptosystems [1], in which each user's public key is his identification information such as his name, address, etc. The systems do not require any key directories. Therefore, identity-based cryptosystems can simplify key management in cryptosystems. Shamir and Fiat proposed identity-based signature schemes [1, 2], and Okamoto proposed an identity-based scheme [3] for a public key distribution system [4]. In these schemes, messages among users are authenticated using each user's identification information.

A two-user secret common key with authentication can be generated by the Shamir scheme [1], Fiat-Shamir scheme [2], Okamoto scheme [3] and so on. If two or more users want to hold a conference, they must derive one common secret communication key for each link in the network. This common key among $m(\geq 2)$ users is called a conference key. Ingemarsson et al. [5] presented a conference key distribution system (CKDS) without authentication, where users are connected in a ring network. The purpose of this paper is to propose an identity-based system for generating a conference key with authentication. We call this system an identity-based conference key distribution system or ICKDS. We propose three types of identity-based conference key distribution systems, which are Type-1 in a ring network [6], Type-2 in a complete graph network [7], and Type-3 in a star network [7].

All ICKDSs are carried out in two phases: the first phase is carried out at a trusted center, and the second phase at each user's location. Once the first phase is carried out, the second phase can be repeated to generate a different conference key. In the second phase, no further interaction with the center is required either to generate a key or to verify proofs of identity. Protocols of Type-1, Type-2, and Type-3 for generating a conference key among m users are described in Sections 2, 3, and 4. Security and transmission efficiency for these protocols are discussed in Sections 5 and 6.

2. ID-based CKDS in a ring (Type-1)

2.1 First phase

During the first phase of Type-1, the center generates two large primes p, q and the product $n = pq$. It determines integers $(e,\ d)$ in a way same as that of the RSA cryptosystem [8]:

$$ed \equiv 1 \pmod{L}, \qquad L = \mathrm{lcm}\ ((p-1),\ (q-1)), \qquad 3 \leq e,\ d < L, \tag{2.1}$$

where lcm denotes a least common multiplier and e is coprime to L. It also determines a prime c $(3 \leq c < L)$, and an integer g which is a primitive element over both $GF(p)$ and $GF(q)$. Let M be the largest number of expected conference members. For user i whose identification information is I_i, the center calculates integers S_i:

$$S_i = I_i^h \bmod n, \qquad h = d^{M-1} \bmod L. \tag{2.2}$$

Note that $I_i = {S_i}^{e^{M-1}} \bmod n$. Finally the center issues a smart card to user i after properly checking his physical identity. This smart card includes the set of integers $(n,\ g,\ e,\ c,\ S_i)$. If no new users are expected, the center can abort numbers p, q and d after all of the data is distributed. Hence, p, q, and d are kept secret from all users, S_i is known only to user i, and n, g, e, c are public and common to all the users.

2.2 Second phase

During the second phase of Type-1, the conference key is generated and simultaneously distributed among $m(\leq M)$ users. Users are connected in a ring so that user i always sends messages to user $i+1$ and user m sends to user 1. For simplicity, an expression of the user label is interpreted as modulo m so that it is between 1 and m. The key generation algorithm is the same for each user. Therefore, it is sufficient to describe the procedure for one user, labeled i, as follows:

step 1: User i generates a random number R_i, and keeps it secret. He sends $(X_i,\ Y_i,\ Z_i)$:

$$X_i = g^{eR_i} \bmod n, \tag{2.3}$$

$$Y_i = S_i g^{cR_i} \bmod n, \tag{2.4}$$

$$Z_i = 1, \tag{2.5}$$

to user $i+1$

step j $(2 \leq j \leq m-1)$: User i receives $(X_{i-1},\ Y_{i-1},\ Z_{i-1})$. He computes T_{i-1}:

$$T_{i-1} = X_{i-1} Z_{i-1}^{e} \bmod n. \tag{2.6}$$

He checks whether the following congruence holds:

$$(Y_{i-1}^{e}/T_{i-1}^{c})^{e^{M-j}} \equiv \prod_{1 \leq k \leq j-1} I_{i-k} \pmod{n}. \tag{2.7}$$

If this check succeeds, user i can verify that the message came via user $i-1$, user $i-2$, ..., and user $i-j+1$ successively. He then sends $(X_i,\ Y_i,\ Z_i)$:

$$X_i = X_{i-1}^{eR_i} \bmod n, \tag{2.8}$$

$$Y_i = Y_{i-1}^{e} S_i^{e^{j-1}} X_{i-1}^{cR_i} \bmod n, \tag{2.9}$$

$$Z_i = T_{i-1}, \tag{2.10}$$

to user $i+1$. Then he proceeds to the next step $j+1$.

step m: User i receives $(X_{i-1},\ Y_{i-1},\ Z_{i-1})$. He computes T_{i-1} by (2.6). He checks whether (2.7) for $j = m$ holds. If the check succeeds, user i can verify that the message came via user $i-1$, user $i-2$, ..., and user $i-m+1$ successively. He then computes conference key K:

$$K = X_{i-1}^{R_i} \bmod n. \tag{2.11}$$

The value of K is the same for all users, because

$$K = g^{e^{m-1}R_1R_2 \ldots R_m} \bmod n. \tag{2.12}$$

2.3 Example of Type-1

Let $m = 4$ and $M = 10$. Transmitted messages $(X_1,\ Y_1,\ Z_1)$ from user 1 to user 2 at intervals of each step are as follows:

$$(g^{eR_1},\ S_1 g^{cR_1},\ 1), \qquad \text{after step 1;}$$

$$(g^{e^2R_4R_1},\ S_4^e g^{ceR_4} S_1^e g^{ceR_4R_1},\ g^{eR_4}), \qquad \text{after step 2;}$$

$$(g^{e^3R_3R_4R_1},\ S_3^{e^2} g^{ce^2R_3} S_4^{e^2} g^{ce^2R_3R_4} S_1^{e^2} g^{ce^2R_3R_4R_1},\ g^{e^2R_3} g^{e^2R_3R_4}), \quad \text{after step 3.}$$

User 1 authenticates the messages if the following congruences hold at each step:

$$(Y_4^e/T_4^c)^{e^8} \equiv (S_4^e g^{ceR_4}/g^{ceR_4})^{e^8} \equiv S_4^{e^9} \equiv I_4 \pmod n, \qquad \text{at step 2;}$$

$$(Y_4^e/T_4^c)^{e^7} \equiv (S_3^{e^2} g^{ce^2R_3} S_4^{e^2} g^{ce^2R_3R_4}/g^{ce^2R_3} g^{ce^2R_3R_4})^{e^7}$$

$$\equiv S_3^{e^9} S_4^{e^9} \equiv I_3 I_4 \pmod n, \qquad \text{at step 3;}$$

$$(Y_4^e/T_4^c)^{e^6} \equiv (S_2^{e^3} g^{ce^3R_2} S_3^{e^3} g^{ce^3R_2R_3} S_4^{e^3} g^{ce^3R_2R_3R_4}/g^{ce^3R_2} g^{ce^3R_2R_3} g^{ce^3R_2R_3R_4})^{e^6}$$

$$\equiv S_2^{e^9} S_3^{e^9} S_4^{e^9} \equiv I_2 I_3 I_4 \pmod n, \qquad \text{at step 4.}$$

Finally he obtains conference key K:

$$K = X_4^{R_1} = g^{e^3 R_1 R_2 R_3 R_4}.$$

3. ID-based CKDS in a complete graph (Type-2)

3.1 First phase

During the first phase of Type-2, the center generates three large primes p, q, and r, and the partial product $n = pq$. It determines integers $(e,\ d)$ in a way similar to that of the RSA cryptosystem:

$$ed \equiv 1 \pmod L, \qquad L = \text{lcm}\,((p-1),\ (q-1),\ (r-1)), \qquad 3 \le e,\ d < L, \tag{3.1}$$

where e is coprime to L. It also determines a prime c $(3 \le c < L)$, and an integer g which is a primitive element over $GF(p)$, $GF(q)$, and $GF(r)$. For user i whose identification information is I_i, the center calculates integer S_i:

$$S_i = I_i^d \bmod nr. \tag{3.2}$$

Note that $I_i = S_i^e \bmod nr$. Finally the center gives the set of integers $(n,\ r,\ g,\ e,\ c,\ S_i)$ to user i in a way similar to that of Type-1. If no new users are expected, the center can abort numbers

p, q and d after all of the data is distributed. Hence, p, q, and d are kept secret from all users, S_i is known only to user i, and n, r, g, e, c are public and common to all the users.

3.2 Second phase

During the second phase of Type-2, the conference key is generated and simultaneously distributed among m users. Users are connected in a complete graph network so that they always send messages to all other users. The key generation algorithm is the same for each user. For convenience, the procedure for two typical users, labeled i and j ($1 \leq i,\ j \leq m,\ i \neq j$), can be described as follows:

step 1: User j generates a random number U_j, and sends E_j:

$$E_j = g^{eU_j} \bmod n \tag{3.3}$$

to user i. He keeps U_j secret.

step 2: User i generates a random number P_i that is coprime to $(r-1)$. He computes $\overline{P}_i$:

$$P_i\overline{P}_i \equiv 1 \pmod{(r-1)}, \tag{3.4}$$

and keeps P_i and $\overline{P}_i$ secret. He generates a random number V_i and keeps it secret. He then sends $(X_i,\ Y_i,\ Z_{ij},\ F_i)$:

$$X_i = g^{eP_i} \bmod nr, \tag{3.5}$$

$$Y_i = S_i g^{cP_i} \bmod nr, \tag{3.6}$$

$$Z_{ij} = E_j^{P_i} \bmod n, \tag{3.7}$$

$$F_i = X_i^{eV_i} \bmod n, \tag{3.8}$$

to user j.

step 3: User j receives $(X_i,\ Y_i,\ Z_{ij},\ F_i)$. He checks whether the following $2(m-1)$ congruences hold:

$$Y_i^e / X_i^c \equiv I_i \pmod{nr}, \tag{3.9}$$

$$Z_{ij} \equiv X_i^{U_j} \pmod{n}. \tag{3.10}$$

If (3.9) and (3.10) hold, user j can verify that the message came from user i. User j generates a secret random number R_j. He then sends $(A_{ji},\ B_{ji},\ C_{ji})$:

$$A_{ji} = X_i^{eR_j} \bmod nr, \tag{3.11}$$

$$B_{ji} = S_j X_i^{cR_j} \bmod nr, \tag{3.12}$$

$$C_{ji} = F_i^{R_j} \bmod n, \tag{3.13}$$

to user i. He keeps $A_{jj} = X_j^{eR_j}$.

step 4: User i receives $(A_{ji},\ B_{ji},\ C_{ji})$. He checks whether the following $2(m-1)$ congruences hold:

$$B_{ji}^e / A_{ji}^c \equiv I_j \pmod{nr}, \tag{3.14}$$

$$C_{ji} \equiv A_{ji}^{V_i} \pmod{n}. \tag{3.15}$$

If (3.14) and (3.15) hold, user i can verify that the message came from user j. He then computes conference key K:

$$K = (\prod_{j=1}^{m} A_{ji})^{\overline{P}_i} \bmod r. \tag{3.16}$$

The value of K is the same for all users, because

$$K = g^{e^2(P_iR_1+P_iR_2+\ldots+P_iR_m)\overline{P}_i} \bmod r = g^{e^2(R_1+R_2+\ldots+R_m)} \bmod r. \tag{3.17}$$

4. ID-based CKDS in a star (Type-3)

Type-2 can be simplified by restricting the process so that $j = 1$ and $2 \le i \le m$. Therefore, users are connected in a star network so that messages are transmitted between user 1 and user i $(2 \le i \le m)$. In this simplified scheme called Type-3, we assume that user 1 collects and delivers messages. Without loss of generality, this 'center user' can be arbitrarily selected among m users.

The key generation algorithm during the second phase of Type-3 is similar to that of Type-2. Note that user 1 can compute conference key $K = g^{e^2R_1}$ at any time. User i $(2 \le i \le m)$ computes conference key K at step 4 by:

$$K = A_{1i}^{\overline{P}_i} \bmod r. \tag{4.1}$$

The value of K is the same for all users, because

$$K = g^{e^2R_1P_i\overline{P}_i} \bmod r = g^{e^2R_1} \bmod r. \tag{4.2}$$

Note that the value of K in Type-3 is dependent on only user 1's secret key R_1, while the value of K in Type-2 is equally dependent on each user's secret key R_i.

5. Security

The security of the proposed systems is based on the difficulty of deriving secret keys such as $(p,\ q,\ d,\ S_i,\ R_i,\ K)$ in Type-1 and $(p,\ q,\ d,\ S_i,\ U_i,\ V_i,\ P_i,\ \overline{P}_i,\ R_i,\ K)$ in Type-2 and Type-3 from public keys, transmitted messages, and other user's secret keys.

(1) Secrecy of $(p,\ q,\ d,\ S_i)$ in all ICKDSs is based on the difficulty of factoring a large number n. If an opponent were able to factor n, he could then compute d from e and L, and could then obtain S_i from d and I_i.

(2) Secrecy of $(R_i,\ K)$ in Type-1 and $(U_i,\ V_i)$ in Type-2 and Type-3 is based on the difficulty of both factoring a large number n and computing discrete logarithms over $GF(p)$ and $GF(q)$. If an opponent were able to factor $n = pq$, he could then compute $g^{eR_i} \bmod p$ and $g^{eR_i} \bmod q$ in Type-1. Moreover, if he were able to compute discrete logarithms over $GF(p)$ and $GF(q)$, he could then compute R_i from $g^{eR_i} \bmod p$ and $g^{eR_i} \bmod q$, and could then obtain K from R_i in Type-1. Similarly, he could compute U_i and V_i, and could obtain Z_{ji} and C_{ji} for passing the check of (3.10) and (3.15), respectively, in Type-2 and Type-3.

(3) Secrecy of $(P_i,\ \overline{P}_i,\ R_i,\ K)$ in Type-2 and Type-3 is based on the difficulty of computing discrete logarithm over $GF(r)$. Since an opponent can compute $\overline{e}$ such that $e\overline{e} \equiv 1 \pmod{(r-1)}$, he can easily derive $g^{P_i} \bmod r$ from X_i. If an opponent were able to derive $P_i \bmod (r-1)$ from $g^{P_i} \bmod r$, he could then compute $\overline{P}_i$ and could obtain K from $\overline{P}_i$ and A_{ji}.

(4) The best known algorithm for factoring $n = pq$ $(p < q)$ requires a running time of $O(\exp((2 + o(1))\sqrt{\log p \log \log p}))$ [9]. Therefore, the designer can choose the sizes of p and q so as to prevent an impersonation attack in all ICKDSs and to ensure the secrecy of K in Type-1. The best known algorithm for computing the discrete logarithm over $GF(r)$ for any prime r requires a running time of $O(\exp((1 + o(1))\sqrt{\log r \log \log r}))$ [10]. Therefore, the designer can choose the size of r to ensure the secrecy of K in Type-2 and Type-3. From the security viewpoint, the size of p and q should be at least 256 bits long, and the size of r should be at least 512 bits long.

(5) The Type-1 scheme corresponds to the most secure version of Ingemarsson's schemes [5] because K has an exponent degree m in an indeterminate R_i.

(6) If an opponent changes transmitted messages $(X_i,\ Y_i)$ to $(X_i a^e,\ Y_i a^c)$ in all ICKDSs, or $(A_{ji},\ B_{ji})$ to $(A_{ji} b^e,\ B_{ji} b^c)$ in Type-2 and Type-3, he can disturb the key distribution system by bypassing the ID check. As a result of this disturbance, each user obtains a different

conference key. However, user i and user j ($1 \leq i,\ j \leq m,\ i \neq j$) can verify the sameness of their keys K by testing that an encrypted message with one's key is successfully decrypted with other's key, called an encryption-and-decryption test. Therefore, this disturbance is detectable.

If the Shamir-Fiat identity-based signature schemes [1, 2] are used to send messages, the disturbance during the transmission can be detected directly after the transmission. The Shamir-Fiat schemes realize both sender authentication and message authentication, while our proposed schemes realize only sender authentication. In key distribution systems, message authentication can be realized after an encryption-and-decryption test.

In the Shamir-Fiat schemes, user i has to know I_j exactly because it is used as input data in the verification process. In our proposed schemes, even if user i remembers I_j imperfectly, sender authentication is possible because he only checks whether he obtains a reasonable I_j as the output data in the verification process.

(7) In Type-1, user $i-1$ can derive S_i^e mod n from his secret key R_{i-1} and transmitted messages $X_i = g^{eR_i}$ after step 1 and $Y_i = S_{i-1}^e g^{ceR_{i-1}} S_i^e g^{ceR_{i-1}R_i}$ after step 2. Even if user $i-1$ obtains S_i^e mod n, it is difficult to derive S_i mod n from S_i^e mod n. Therefore, user $i-1$ cannot pretend to be user i.

(8) The checks of (3.10) and (3.15) in Type-2 and Type-3 are introduced to detect impersonation attacks using passive and active wiretaps. Since the purpose and function of (3.10) and (3.15) is the same, we describe the case for (3.10) as an example.

If the check of (3.10) and related computations of (3.3) and (3.7) are omitted, an opponent can pretend to be user i and finally obtains K as follows: After the opponent wiretaps (X_i mod nr, Y_i mod nr), he can produce the following ($\widetilde{X}_i$ mod nr, $\widetilde{Y}_i$ mod nr) to pass the check of (3.9):

$$P'\overline{P'} \equiv 1 \pmod{(r-1)}, \qquad e\overline{e} \equiv 1 \pmod{(r-1)}, \tag{5.1}$$

$$X_i' = g^{eP'} \bmod r, \qquad Y_i' = (I_i(X_i')^c)^{\overline{e}} \bmod r, \tag{5.2}$$

$$\begin{cases} \widetilde{X}_i \equiv X_i \bmod n, \\ \widetilde{X}_i \equiv X_i' \bmod r, \end{cases} \qquad \begin{cases} \widetilde{Y}_i \equiv Y_i \bmod n, \\ \widetilde{Y}_i \equiv Y_i' \bmod r, \end{cases} \tag{5.3}$$

He sends ($\widetilde{X}_i$, $\widetilde{Y}_i$) to user j and gets $\widetilde{A}_{ji} = \widetilde{X}_i^{eR_j}$ from user j. From the unfairly generated $\widetilde{A}_{ji}$ and $\overline{P'}$, he computes $\widetilde{A}_{ji}^{\overline{P'}} = g^{e^2R_j}$ and finally obtains K. Note that this impersonation attack is made without the knowledge of S_i mod nr.

If the check of (3.10) is introduced, an opponent cannot pretend to be user i who is absent from a current ICKDS. The opponent generates ($\widetilde{X}_i$, $\widetilde{Y}_i$) from X_i mod nr previously generated by regular user i. However, he cannot produce Z_{ij} to pass the check of (3.10) because he does not know the previous value of P_i that is a parameter of X_i and Z_{ij}. Note that the previously generated Z_{ij} is not reusable because the value of U_j changes at each ICKDS. Therefore, the check of (3.10) detects this impersonation attack.

As pointed out in [7], if an opponent tries to pretend to be user i who is present at the current ICKDS, the check of (3.10) is ineffective. The opponent interrupts transmission from user i, and sends ($\widetilde{X}_i$, $\widetilde{Y}_i$, Z_{ij}) generated from a current (X_i, Y_i, Z_{ij}). Since this Z_{ij} passes the check of (3.10), the opponent can get correct key K. However, regular user i receives $\widetilde{A}_{ji}$ generated from the false P', and he finally obtains false key $\widetilde{K} = g^{e^2(P' \sum_{j=1}^{m} R_j)\overline{P}_i} \bmod r$. In this one-directional attack, the opponent cannot get this false key $\widetilde{K}$. Therefore, the one-directional real time impersonation attack is detectable after the encryption-and-decryption test by regular user i [7].

By extending the above analysis, Yacobi [11] showed a bidirectional real time attack between user i and user j in a star system (Type-3). In his attack using false random number R'_j, the opponent can get the same false key $\widetilde{K}' = g^{e^2 R'_j} \bmod r$ as user i gets. His attack method can be generalized to a complete-graph system (Type-2). Since the opponent can hold both a correct key and a false key, this bidirectional impersonation attack would be successful if the opponent could change all interactions with regular user i after the key generation. If a conference is carried out in radio broadcast networks, this attack would be detectable by the encryption-and-decryption test because it seems to be physically impossible for the opponent to change the radio transmissions.

6. Transmission efficiency

Type-1 requires $(m - 1)$ steps for transmission because messages must be sent sequentially among m users. However, Type-2 and Type-3 requires 3 steps for transmissions for any m because messages can be broadcasted simultaneously. The total numbers of message transmissions among m users in Type-1, Type-2, and Type-3 are $m(m-1)$, $3m(m-1)$, and $3(m-1)$, respectively. The total message length of transmissions in each Type-1, Type-2, and Type-3 is $3m(m - 1) \log n$, $4m(m-1)(\log nr + \log n)$ and $4(m-1)(\log nr + \log n)$ bits, respectively. We compare the above indices for transmission efficiency among each type. For the numbers of sequential steps, Type-2 and Type-3 are better than Type-1 if $m \geq 5$. For the total numbers of message transmissions, Type-3 is better than Type-1 if $m \geq 4$. For the total message length of transmissions, assuming $\log n$=512 and $\log r$=256, Type-3 is better than Type-1 if $m \geq 4$. Summarizing the comparisons from the viewpoint of transmission efficiency, Type-1 is the best if $m = 3$, and Type-3 is the best if

$m \geq 4$. Note that if m=2, then it is the best to use the simpler schemes such as [1, 2, 3]. Expansion of message transmission is needed to ensure the security of a conference key.

Acknowledgement: We would like to thank T. Okamoto of NTT Corporation for his valuable discussions.

References

[1] SHAMIR, A. :'Identity-based cryptosystems and signature schemes', Proceedings of Crypto-84, Santa Barbara, August 1984, pp47-53

[2] FIAT, A. and SHAMIR, A. :'How to prove yourself: Practical solutions to identification and signature problems', Proceedings of Crypto-86, Santa Barbara, August 1986, pp18-1–18-7

[3] OKAMOTO, E.:'Proposal for identity-based key distribution systems', *Electron. Lett.*, 1986, **22**, pp1283-1284

[4] DIFFIE, W., and HELLMAN, M. E. :'New directions in cryptography', *IEEE Trans.* 1976, **IT-22**, pp644-654

[5] INGEMARSSON, I, TANG, D. T. and WONG, C. K. :'A conference key distribution system', *IEEE Trans.* 1982, **IT-28**, pp714-720

[6] KOYAMA, K. :'Identity-based conference key distribution system', *IEE Electron. Lett.*, May 7, 1987, Vol.23, No.10, pp.495-496.

[7] KOYAMA, K. and OHTA, K. :'Identity-based conference key distribution systems in broadcast networks' *IEE Electron. Lett.*, June 4, 1987, Vol.23, No.12, pp.647-649.

[8] RIVEST, R. L., SHAMIR, A., and ADLEMAN, L.:'A method for obtaining digital signatures and public-key cryptosystems', *Commun. ACM,* 1978, 21, pp120-126

[9] LENSTRA, Jr. H. W. :'Factoring integers with elliptic curves', preprint, May 1986

[10] COPPERSMITH, D., ODLYZKO, A. M. and SCHROEPPEL, R. :'Discrete logarithms in GF(p)' *Algorithmica* 1986, **1**, pp1-15

[11] YACOBI, Y. :'Attack on the Koyama-Ohta identity-based key distribution STAR system', private communication, July 13, 1987.

On the KEY PREDISTRIBUTION SYSTEM: A Practical Solution to the Key Distribution Problem

Tsutomu MATSUMOTO
&
Hideki IMAI

Division of Electrical and Computer Engineering
YOKOHAMA NATIONAL UNIVERSITY
156 Tokiwadai, Hodogaya, Yokohama, 240 Japan

August 1987

Abstract To utilize the common-key encryption for the efficient message protection in a large communication network, it is desired to settle the problem of how to distribute the common keys. This paper describes a practical solution called the ***key predistribution system*** (KPS, for short), which has been proposed by the present authors. On request, the KPS quickly brings a common key to an arbitrary group of entities in a network. Using the KPS, it is quite easy to construct an enciphered one-way communication system, as well as an enciphered two-way (interactive) communication system. For example, even in a very large public network, the KPS can be applied to realize a practical enciphered electronic mailing service directed to individuals. This paper presents secure and efficient realization schemes for the KPS. This paper also discusses the security issues and the variety of applications of them.

1. Introduction

Cryptography is a key technology for the information security of our everyday lives. Main features of today's and near future's cryptographic communications include the point that they are performed in large open networks having a lot of *entities* (or, users, subscribers, terminals, ...): we will often face to the need of cryptographic communications with what we don't know very well, in, for examples, facsimile systems, portable telephone systems, personal (card) computer networks, or, ...

For such large-network cases, both common-key and public-key approaches have had to require somewhat complex and costly protocols to overcome the so-called key distribution problem. And it has not been so easy to build a practical enciphered electronic mailing system for a large network by applying those conventional approaches.

Main purposes of this paper are to introduce the ***key predistribution system*** (KPS, for short) proposed by the present authors in [MI86a,b,c] and to present secure and efficient realization schemes for the KPS. The KPS is a kind of key distribution system for the common-key systems and, on request, brings a common key to each member of any group of specified entities in a network, without previous communications among the group nor accesses to any public key directory or whatsoever. So, the KPS has the variety of applications which have never been realized by the conventional common-key nor public-key systems alone.

In the following, the definition, the applications, and the security of the KPS are described in **2**, in **3**, and in **4**, respectively. Then several realization schemes based on the linear algebra and/or smart cards are proposed in **5**.

2. The Key Predistribution System

Imagine a public communication network consisting of many entities (users), say, *entity*1,*entity*2,...,*entity* n and trusted managing center(s), say, *center*1,*center*2,...,*center* s, where n and s are positive integers. We assume each entity (say, *entity* i) has its own *identity* y_i : the identity y_i may be the entity's name, address, etc., or combinations of such items.

The Key Predistribution System (KPS) is a method of sharing a cryptographic key, among any group of specified entities in such a network, according to the following processes [MI86a,b,c]:

(1) Generation of Center-Algorithms
(2) Calculation and distribution of each entity's Secret-Algorithm
(3) Key sharing among the group.

Process(1) is required only when the system is setting up or renewed. At this process, each center (say, *center* p) independently generates a special Center-Algorithm $\mathbf{G}_p$ which should be kept secret by the owner (*center* p).

Process(2) is operated between an entity (say, *entity* i) and every center only when the entity joins the system or an emergency arises. In Process(2), for *entity* i, each center (say, *center* p) applies its Center-Algorithm $\mathbf{G}_p$ to the identity y_i of *entity* i, and obtains an algorithm $X_{pi} = \mathbf{G}_p(y_i)$, and sends X_{pi} only to *entity* i, with the aid of some cryptographic or physical protection mechanisms, eg., smart cards. Here, a smart card means a well-access controlled physically secure small computing device. *Entity* i combines the received algorithms $X_{1i}, X_{2i}, \ldots, X_{si}$ and keeps the resulting Secret-Algorithm X_i confidentially. For the convenience of *entity* i, the the Secret-Algorithm X_i may be generated by some interactive procedures among the centers and a smart card owned by *entity* i.

If Process(1) and Process(2) have been set up, a group of entities can operate Process(3) to have a common cryptographic key whenever they like. Each member of the group computes the same key by inputting into its Secret-Algorithm the identities of all members except itself. Let $X_i^{(e)}$ denote the Secret-Algorithm of *entity* i used for key sharing among a group of e entities ($e \geq 2$). Then, for example, if *entityA* and *entityB* want to share a cryptographic key k, both entities compute k in the following manner:

$$k = X_A^{(2)}(y_B), \quad k = X_B^{(2)}(y_A).$$

If three entities *entityA*, *entityB*, and *entityC* want to have the same cryptographic key l, they obtain l as follows:

$$l = X_A^{(3)}(y_B, y_C), \quad l = X_B^{(3)}(y_C, y_A), \quad l = X_C^{(3)}(y_A, y_B).$$

In the key predistribution system, once the prerequisite Process(1) and Process(2) are accomplished, each entity does not have to communicate with any centers nor any entities for the purpose of acquiring cryptographic keys. The only necessary items are the other member's identities. This is the most significant feature of the KPS.

Such an advantage could not be attained by any conventional cryptographic methods in a large communication network. Indeed, the key translation / distribution center methods [ISO87], which seem to be very popular, require extra communications on channels with confidentiality and authenticity. And even for the public-key distribution / encryption / signature systems [DH76], extra authentic communications are indispensable. They may be maintained by the central public-key-file manager and/or the use of certified public keys based on some digital signature schemes.

This feature of the KPS makes it possible to readily build an one-way end-to-end encryption system even in a very large network. That is, without extra crypto-processing in the network, one can send enciphered electronic mails to anybody who joins the system. The target of the mail may consist of several entities. Of course, the KPS is still effective for small networks and for enciphered two-way (interactive) communications such as telephones.

For the KPS, only what distinguish the entities are the identities of them. So, depending on the configuration of the identities, the KPS can provide a variety of key sharings. And these identities are the only public items used in the KPS. Note that the Center-Algorithms and the Secret-Algorithms are kept secret by their owners.

Remark The name "Key Predistribution System" comes from the fact that the notion of KPS includes the following primitive method: in Process(1) each center generates keys for all possible groups, then in Process(2) each entity receives from the centers a list of all keys for the groups including the entity as a member, and in Process(3) each member of a group reads out the common key from its list according to the identities of the group members. It is apparent that this method is secure as long as the centers are trusted. But for large networks, this primitive approach is impractical since the memory size of the list becomes extraordinary big. This suggests that one of the points we must attention in the research of the KPS is how to reduce the memory size and the computational complexity of each entity's Secret Algorithm with keeping enough security level.

Remark In the point that the identities play important roles, the notion of KPS is like the ID-based cryptosystem; the independent work of A. Shamir's [S84]. But we believe that the KPS is the first unified treatment of the method of key sharing among a group of entities without previous communications.

How to construct the KPS ? The following is a very general answer. Select an e-input symmetric function g. Then, for each identity y_i generate a certain $(e-1)$-input algorithm $X_i^{(e)}$ satisfying

$$X_i^{(e)}(\xi_1, \ldots, \xi_{e-1}) = g(y_i, \xi_1, \ldots, \xi_{e-1}).$$

In **5** we will construct some schemes by adopting e-linear mappings as the g in this idea. And we feel that there are other useful schemes for the KPS constructed by some combinatoric or geometric or algebraic approaches.

3. Applications of the KPS

The key predistribution system has wide applications. Indeed it can be effectively used for any fields to which the common-key systems can be applied. Therefore, some combinations of the KPS and common-key cryptographic techniques can bring the functions including the keeping confidentiality and authenticity of messages, the peer entity authentication, the access control supporting, etc.

Moreover, we believe that the KPS has many potential applications to the smart-card-based systems, since the KPS can be elegantly realized even by today's not-so-powerful smart cards, as we will demonstrate later, and since the KPS seems to be the best fit cryptographic partner for the smart cards.

In the following we introduce some of these applications.

For $i = 1, 2$, let (E_i, D_i) be a common-key cryptosystem, i.e., E_i, D_i be a pair of algorithms satisfying

$$D_i(k; E_i(k; M)) = M,$$

for any key k and any message M. For (E_i, D_i), fast and strong stream/block algorithms are suitable for practical purposes.

Let (H, U, V) be an integrity mechanism, i.e., H, U, V be a triple of algorithms such that V takes either OK or NG and that for any message M there hold

$$U(H(M)) = M, \quad V(H(M)) = OK$$

and that if $N = H(M)$ is altered into some N' then in high probability,

$$V(N') = NG.$$

An error-detecting code can be used as (H, U, V).

Electronic Mail: Suppose that *entityA* wants to send *entityB* a message M in an enciphered form.

(Step 1) $\{entityA\}$: Generate a random number r. Compute

$$F = E_1(X_A^{(2)}(y_B); r)$$

$$C = E_2(r; H(M))$$

and send the mail $(y_B; y_A; F; C)$ to *entityB* or to the *entityB*'s mail box.

(Step 2) $\{entityB\}$: Receive $(y_B; y_A; F'; C')$ from *entityA* or from the mail box. Compute

$$r' = D_1(X_B^{(2)}(y_A); F')$$

$$N' = D_2(r'; C')$$

$$M' = U(N')$$

$$T = V(N')$$

and judge that $M' = M$ and the sender is really *entityA* iff $T = OK$.

Electronic Broadcasting Mail: Suppose that *entityA* wants to send *entityB*1, *entityB*2, ..., *entityBe* a message M in an enciphered form.

There are two generalizations of above Electronic Mail.

One is to utilize $X_A^{(e+1)}$ instead of $X_A^{(2)}$. That is, *entityA* broadcasts the mail

$$(y_{B1}, y_{B2}, \ldots, y_{Be}; y_A; F^*; C)$$

where $F^* = E_1(X_A^{(e+1)}(y_{B1}, y_{B2}, \ldots, y_{Be}); r)$.

The other is to use F_j for *entityBj* for $j = 1, \ldots, e$. That is, in this method *entityA* broadcasts the mail

$$(y_{B1}, y_{B2}, \ldots, y_{Be}; y_A; F_1, F_2, \ldots, F_e; C)$$

where $F_j = E_1(X_A^{(2)}(y_{Bj}); r)$, and *entityBj* does the similar procedure in (Step 2).

The advantages of the former method are that the length of the mail is independent from the number of the receivers and that each receiver can check whether the mail was really directed to the specified receivers. On the other hand, the memory size and the computational complexity of Secret-Algorithm $X_A^{(e+1)}$ may rapidly increase according to $e+1$. Opposed to this, although the increment of the mail length is linear in e, the latter method requires relatively cheap Secret-Algorithm $X_A^{(2)}$ only. Especially, the load of each receiver is not affected by the increase of receivers. We think that the latter method is much practical than the former.

Remark In the above mailing methods, no crypto-processing is required on the mails transferred through the network. Thus those enciphered electronic mails can be treated just as usual cleartext mails. This means the readiness of implementation.

Remark Since the above mailing methods are based on common-key systems, sender authentication is naturally implemented. This is an advantage. On the other hand, there may be a case where the sender wants to send an anonymous message. To attain this end, it is sufficient to prepare another identity (pen name) and the corresponding Secret-Algorithm.

Implementation with Smart Cards and Terminals: If Secret-Algorithms and other crypto-algorithms are stored and worked in their owner's powerful smart cards, there is no problem. We do hope that such powerful and compact devices will be available in near future. However, since at present any smart card is not so powerful, it seems to be practical to implement the KPS and related crypto-functions by some combinations of smart cards and terminals.

Smart cards are assumed to have Secret-Algorithms and some private informations, but not to have enough computing power. Terminals are assumed to have enough computing ability. One of the problems is how to work the Secret-Algorithm using both facilities so that the smart card obtains the key but that the terminal cannot get it. An example of solutions will be given in **5.4**.

Another problem is how to efficiently do the encipherment and the decipherment by using the smart card having the key and the terminal, so that the key cannot be known to the terminal. For the case of above mentioned Electronic (Broadcasting) Mail, one of the possible solutions is to implement the algorithms E_1 and D_1 and the random number generation in the smart cards, and to implement (E_2, D_2) and (H, U, V) in the terminals.

4. Security of the KPS

The security of the KPS highly depends on the centers. If there is only one center, it can do everything since it can readily compute any key for any group in the network. But if we pose several centers, none of keys can be disclosed by the center side as long as there is at least one trusted center.

In Process(2) of the KPS, the Secret-Algorithm for *entity i* should be passed only to *entity i*. Thus rigorous entity identification must be adopted. But this does not mean the necessity of individual identification. As we have mentioned, entities are distinguished only by their identities.

Generally speaking, in any key predistribution system, information or knowledge on the Center-Algorithm $\mathbf{G}_p$ is distributed into the Secret-Algorithms $X_1, X_2, \ldots, X_n$. Therefore, if enough number of entities (breakers) collaborate, they may obtain important information which enables to completely or partially determine the keys shared by other groups.

Of course, if the center side embed the Secret-Algorithm in a well access-controlled physically secure computing device and pass it to its owner (an entity), then the Secret-Algorithm can be used for key sharing while it cannot be read even by its owner. Thus if complete physical security is available, any key predistribution system is secure. But, today, it is rather difficult to expect complete physical security. For example, a smart card, one of the hopeful candidates for such devices, does not seem to have complete physical security.

So in practical situations, the KPS should be constructed so that valuable information on $\mathbf{G}_p$ or X_t cannot be derived unless the breakers includes a lot of entities or unless a huge amount of computation is completed. We refer to the former as the information-theoretic security and the latter as the complexity-theoretic security.

5. Linear Schemes for the KPS

In this section, we present a class of realization schemes for the KPS. Its name is the *linear schemes*. The information-theoretic security of a linear scheme is reduced to the linear-independency of a certain set of vectors.

5.1 Definition of Linear Schemes

To simplify the description, we assume here that there is only one center in the network considered. For the general case, see [MI86d].

Let q be a prime power and m, h be positive integers. Let $Q = GF(q)$ and Q^m denote the vector space consisting of all m-row vectors over Q.

Assume that each entity's (say, *entity i*'s) identity y_i belongs to a set I and that $y_i \neq y_j$ if $i \neq j$.

And let R denote an one-way algorithm implementing an injection from I to Q^m.

The *center* selects (m,m) symmetric matrices $G_1, G_2, \ldots, G_h$ over Q randomly and independently from other entities.

The *center* generates the Center-Algorithm G which generates the algorithm X_i :

$$X_i(\xi) = x_i R(\xi)^T, \quad \xi \in I$$

for each $y_i \in I$. Here, x_i is an (h,m) matrix defined by

$$x_i{}^T = [G_1 R(y_i)^T, \ldots, G_h R(y_i)^T].$$

Each entity (say, *entity i*) receives its own Secret-Algorithm X_i from the *center*.

If *entityA* and *entityB* want to share a common cryptographic key, *entityA* computes $X_A(y_B)$ and *entityB* computes $X_B(y_A)$ independently. They are h-vectors over Q. It is simple to check that both vectors are the same.

If we use symmetric e-linear mappings instead of the above mentioned G_l's (symmetric bilinear mappings), we have similar schemes for key sharing among e entities.

5.2 Security of Linear Schemes

Since we assumed that there is only one center in the network, the *center* must be trusted.

To completely break the linear scheme is to determine the matrix

$$G = [G_1, \ldots, G_h].$$

By the definition of matrix x_i, at least *rank* G entities should cooperate to completely determine G. The value of *rank* G is approximately m in high probability.

Next we discuss the possibility of determining other Secret-Algorithms by some entities.

Let E denote the set of all entities in the system. It is readily derived that "even if all entities in a subset E_B of E cooperate and use $\{x_j \mid j \in E_B\}$, they cannot have any valuable information on determining a key for arbitrary pair of entities in $E - E_B$" is equivalent to "for each $i \in E - E_B$, $R(y_i)$ is linearly independent from $\{R(y_j) \mid j \in E_B\}$". Thus the information-theoretic security of the linear scheme is reduced to the linear-independency of the vector set

$$U = \{R(y_i) \mid i \in E\}.$$

So, there is a tight relation between the linear schemes and the theory of linear error-correcting codes. Consult [MI86d] for more precise discussions.

One direction is to use well known algebraic or algebraic-geometric codes. But this approach becomes rather costly when we use a big security parameter m. The other direction is to utilize random codes. By using a technique similar to one for deriving the famous Gilvert-Varsharmov bound, we can see that almost all algorithms R bring good codes and good U for our purpose. Thus we prefer to use one-way algorithm R which runs very quickly. Candidates of R may include the iterated DES-like algorithms.

The one-way property of R is required for increasing the computational load of searching the members and the targets of the breakers.

5.3 Features of Linear Schemes

The greatest features of the linear schemes are their simplicity and efficiency. Indeed, the memory size of each Secret-Algorithm is the sum of $hm \log_2 q$ $[bit]$ and the memory size of R. And the computational complexity of each Secret-Algorithm is the sum of $O(hm)$ $[Q - operation]$ and the complexity of R. The memory size and complexity of R can be bounded sufficiently low if we use R suggested in **5.2**.

The class of linear schemes is wider than its appearance. For example, if we select (multi-variate) polynomial functions instead of the e-linear mappings, we cannot extend the class. The reasons are that any polynomials are rewritten as linear polynomials by some translations of the set of indeterminates and that such translations can be absorbed in the R. Moreover some scheme can be interpreted as the composition of a linear scheme and some back-end algorithm such as the discrete exponentiation.

5.4 Practical Linear Schemes with Smart Cards

In this subsection we try to demonstrate the usefulness of the linear schemes by an example. Let $m = lh$ and divide x_A as

$$x_A = [x_{A1}, \ldots, x_{Al}],$$

where x_{Aj}'s are (h, h) matrices. Select permutation matrices $P_{A1}, \ldots, P_{Al}$ and a non-singular matrix E_A and compute

$$V_{Aj} = P_{Aj}^{-1} E_A^{-1} x_{Aj}$$

for $j = 1, \ldots, l$.

We divide the Secret-Algorithm for *entityA* into two parts. One is a set of R and V_{Aj}'s stored in a memory card, which is non-intelligent but has relatively large memory capacity. The other part is a set of E_A and P_{Aj}'s stored in a smart card, which is physically protected and has certain computing ability.

When *entityA* use these, R and V_{Aj}'s will be loaded into its terminal, which may be its personal computer, and performed in the terminal. That is, if *entityA* enters some identity, say y_B into the terminal, then the terminal applies R to the input and computes l h-vectors s_j and then multiplies V_{Aj}'s to s_j's and and outputs the resulting h-vectors t_j.

These outputs are transferred into the smart card to be processed into the common key. In the smart card, every t_j is at first permuted according to P_{Aj} then added into a single h-vector u, which is then multiplied by E_A into the h-vector key k.

This method succeeds in reducing the load of the smart card with keeping the secrecy of the Secret-Algorithm from the terminal.

When $q = 2$, $l = 128$, and $h = 64$, the required size of the memory card is 512 $Kbits$ + (some for R) and that of the smart card is 52 $Kbits$. In this case, we can have a KPS by which an h-bit key is brought to any pair of entities in a network consisting of up to 10^{12} entities. The system is primarily protected by the physical security of each smart card. And the system cannot be completely broken unless at least 8192 entities break their own smart cards and collaborate with each other. And any breakers (collaboration of entities which have known their own Secret-Algorithms) cannot obtain any keys for the entities outside of the breakers, unless the number of members is greater than about 256.

References

[DH76] Diffie,W. and Hellman,M.E.,"New directions in cryptology," *IEEE Trans. on Information Theory*, Vol. IT-22, No. 6, November 1976, pp.644-654.

[ISO87] ISO/TC68, "Banking-Key management (wholesale)," ISO/DIS 8732, February 1987.

[MI86a] Matsumoto,T. and Imai,H., *Patent application*, July, 1986.

[MI86b] Matsumoto,T. and Imai,H., "The Third Key Distribution System," *Proceedings of the 1986 Workshop on Cryptography and Information Security, Yokohama, Japan*, August 27, 1986, pp.39-41.

[MI86c] Matsumoto,T. and Imai,H., "The Key Predistribution System," *IECE Technical Report* TGIT86-54, Institute of Electronics and Communications Engineers of Japan, Vol.86, No.145, September 18, 1986, pp.29-34.

[MI86d] Matsumoto,T. and Imai,H., "A Key Predistribution System Based on Linear Algebra," *Proceedings of the* 9th *Symposium on Information Theory and Its Applications, Akakura, Japan*, October 29, 1986, pp.713-718.

[S84] Shamir,A., "Identity-Based Cryptosystems and Signature Schemes," *Advances in Cryptology: Proceedings of CRYPTO84*, Springer LNCS 196, 1985, pp.47-53.

Key Distribution Systems Based on Identification Information

Eiji OKAMOTO

Information Basic Research Laboratory
C&C Information Technology Research Laboratories
NEC Corporation
4-1-1 Miyazaki, Miyamae-ku
Kawasaki, Kanagawa, 213, Japan
Telephone: +81-44-855-1111, Ext. 3631

ABSTRACT

Two types of key distribution systems based on identification information are proposed, one for decentralized networks and the other for centralized networks. The system suitable for decentralized networks is analogous to the Diffie-Hellman public key distribution system, in which the former uses each user's identification information instead of a public file used in the latter. The proposed system is able to defend the networks from impostors. The system suitable for centralized networks, which is less analogous to the Diffie-Hellman system, can also defend the networks from impostors, though the network center does not have any directory of public or private key-encrypting keys of the users. Both of the systems proposed in this paper do not require any services of a center to distribute work keys, or users to keep directories of key-encrypting keys. Therefore, key management in cryptosystems can be practical and simplified by adopting the identity-based key distribution systems.

1. Introduction

Diffie and Hellman first proposed the idea of the public key distribution system in which two users, knowing only public information, could establish a secret key for conventional cryptosystems[1]. However, in order to establish a correct key, it requires an authenticated file of user's public information.

This paper proposes two types of key distribution systems based on identification information, one for decentralized networks[2] and the other for centralized networks. The system suitable for decentralized networks is analogous to the Diffie-Hellman public key distribution system, in which the former uses each user's identification information instead of a public file used in the latter. Any information such as one's name and address can be used as the identity information if it is known to everybody and uniquely identifies the user. The proposed system is able to defend the networks from impostors. The system suitable for centralized networks, which is less analogous to the Diffie-Hellman system, can also defend the networks from impostors, though the only network center uses the identification information. Both of the systems proposed in this paper do not require any services of a center to distribute work keys, or users to keep directories of key-encrypting keys.

The identity-based key distribution systems assume the existence of a trusted card issuer. The only purpose of the center is to give each user a personalized IC card, when the network is initially set up or a new user joins the network. Even when a new user joins, previously delivered cards to other users need not be updated.

The advantages of applying identification information to cryptosystems have been already shown by Shamir at Crypto'84[3]. We applied his idea to the Diffie-Hellman public key distribution system to eliminate the public file.

Key distribution systems using the public key certificates[4], which are user's public keys signed by a central authority, have almost same advantages. However, the identity-based key distribution systems are simpler than the certificate systems whatever public key cryptosystem is used for the certificates. When RSA public key cryptosystem[5] is used, all certificates must have different modulus n[6], whereas the modulus n in the proposed systems is a constant. Since no other public key cryptosystems have simpler form than RSA, the proposed systems are considered to be more practical than the certificate systems.

2. The Principle of Identity-based Key Distribution Systems

The identity-based key distribution systems have two phases: card issue phase and key generation phase. On the request of a new user, the card issuer gives him his card if he is confirmed. Each user with his own IC card can generate a work key through the protocols below. The IC cards should not be rewritten but by the card issuer.

2.1 The System Suitable for Decentralized Networks

In decentralized networks, there are no network centers, and each user communicates with other users directly. In these networks, the card issuer generates two prime numbers p and q about 256 bits long, and determines numbers e and d, satisfying

$$e \cdot d (\mathrm{mod}\,(p-1) \cdot (q-1)) = 1, \qquad (1)$$

with both e and d less than $n = p \cdot q$. It also determines an integer g which is a primitive element in GF(p) and GF(q). For user i whose identification information is IDi, the center calculates an integer si:

$$si = IDi^{-d} \ (\mathrm{mod}\ n), \qquad (2)$$

and stores the set of integers (n,g,e,si) in his IC card, and gives it to him. Number d can be aborted after all the cards are distributed. If there are no new users expected, even the center can be closed. Hence, d is kept secret from any user, si is known only to user i and n,g,e are common to all the users. Figure 1 (a) illustrates the card issue phase.

When users i and j wish to obtain a work key between themselves, user i generates a random number ri and sends user j the integer xi:

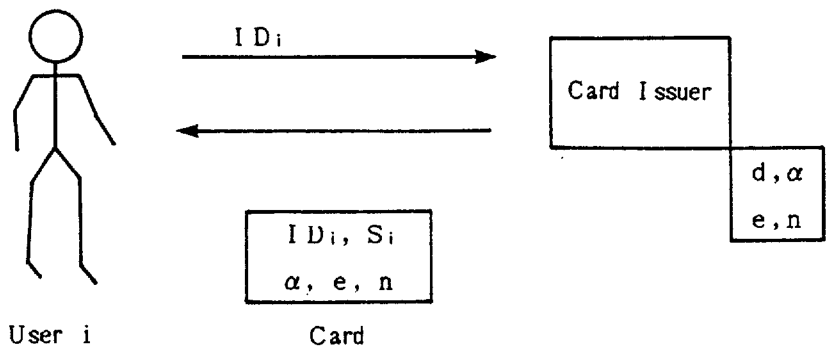
$$xi = si \cdot g^{ri} \ (\mathrm{mod}\ n). \qquad (3)$$

User j also generates a random number rj and sends user i the integer xj:

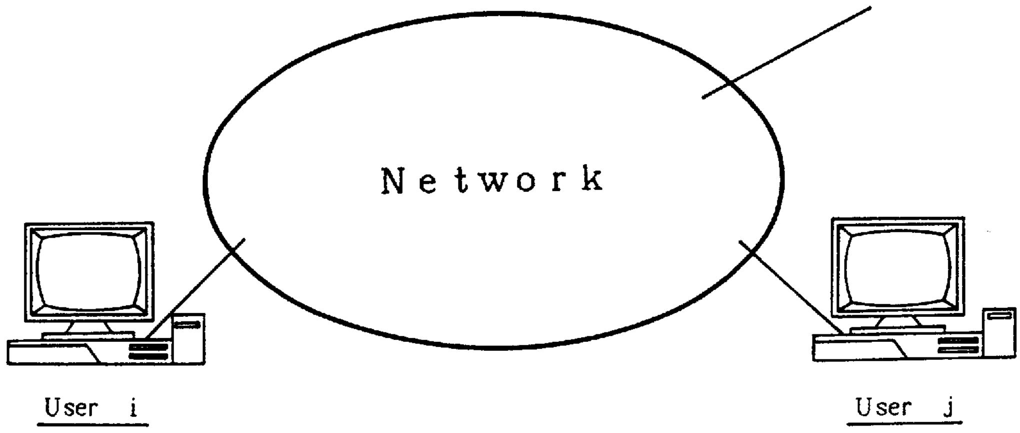
$$xj = sj \cdot g^{rj} \ (\mathrm{mod}\ n). \qquad (4)$$

Then, users i and j compute work keys WKi,WKj respectively as follows:

$$WKi = (xj^{e} \cdot IDj)^{ri} \ (\mathrm{mod}\ n) \qquad (5)$$



(a) Card Issue Phase



Random Number r_i

$x_i = S_i \cdot \alpha^{r_i} \pmod n$

$wk_i = (x_j^{e} \cdot ID_j)^{r_i} \pmod n$

Random Number r_j

$x_j = S_j \cdot \alpha^{r_j} \pmod n$

$wk_j = (x_i^{e} \cdot ID_i)^{r_j} \pmod n$

(b) Key Generation Phase

Fig. 1. Identity-based Key Distribution System
(Decentralized Network)

$$WKj=(xi^{e}\cdot IDi)^{rj} \pmod{n}. \tag{6}$$

Both work keys turn out to be equal, because

$$WKi=WKj=g^{e\cdot ri\cdot rj} \pmod{n}. \tag{7}$$

Figure 1 (b) shows the key generation phase.

2.2 The System Suitable for Centralized Networks

In centralized networks, each user can communicate with other users only via a network center. In these networks, the card issuer gives (n,e,r) to the network center, and distributes each IC card storing (n,g,y,si) to user i, where n,g and si have the same properties as in 2.1, r is any fixed integer less than n, and y is

$$y=g^{e\cdot r} \pmod{n}. \tag{8}$$

Figure 2 (a) illustrates the card issue phase.

When user i wishes to generate a work key between him and the network center, he generates a random number ri and sends xi satisfying

$$xi=si\cdot g^{ri} \pmod{n} \tag{9}$$

to the network center. The work key WKi is given by

$$WKi=y^{ri} \pmod{n}. \tag{10}$$

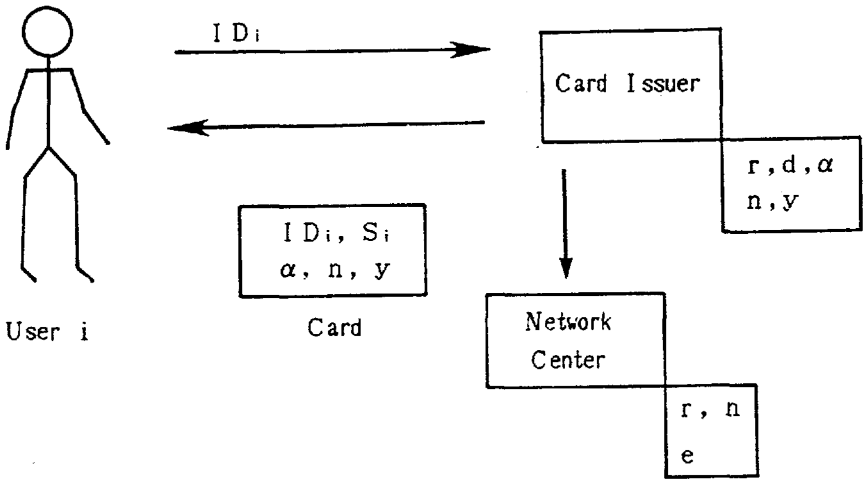
Receiving xi from user i, the center calculates the work key WK as follows:

$$WK=(xi^{e}\cdot IDi)^{r} \pmod{n}. \tag{11}$$

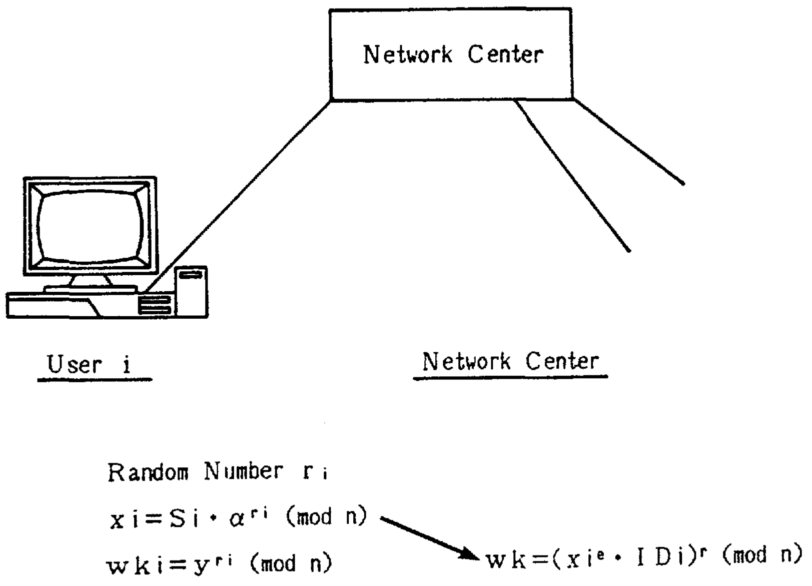
Both work keys WKi and WK are equal to

$$WKi=WK=g^{e\cdot ri\cdot r} \pmod{n}. \tag{12}$$

Figure 2 shows the key generation phase.



(a) Card Issue Phase



(b) Key Generation Phase

Fig. 2. Identity-based Key Distribution System (Centralized Network)

3. Security of the Identity-based Key Distribution Systems

The security of the systems depend on the difficulty of finding x and r satisfying

$$IDk \cdot x^{e} \equiv g^{e \cdot r} \pmod{n} \tag{13}$$

for some k. If found, one can send the integer x to any user of the decentralized networks or the network center of the centralized network, disguising himself as user k. Finding x in Eq.(13) with r given is equivalent to decrypting RSA cryptosystems without decryption keys. Calculation of r in Eq.(13) with x given becomes the discrete logarithm computation.

Assume there were i and j such that

$$IDk \equiv IDi^{a} \cdot IDj^{b} \pmod{n} \tag{14}$$

for some integers a and b. Then, user i and j can succeed to satisfy Eq.(13) together, with x and r given by

$$x \equiv xi^{a} \cdot xj^{b} \pmod{n} \tag{15}$$

$$r = a \cdot ri + b \cdot rj. \tag{16}$$

However, the probability of the existence of a and b satisfying Eq.(14) can be made small, by introducing redundancy to each ID or applying a hash function to them. Even if the number of ID's in use is one trillion, the ratio of ID to the number less than n is $10^{12}/2^{512} = 10^{-142}$.

4. Applications

The identity-based key distribution system for decentralized networks is applicable to any bidirectional real-time communication networks. Telephone is a typical example. Recently almost all personal computer networks have real-time "talk" services which the proposed system can be applied to.

The centralized type system is applicable to any networks with a network center---for example, an electronic mail system, a data base system, and so on.

When these networks adopt the identity-based key distribution system, the most time-consuming operation is the exponentiation:

$$y=x^{e} \pmod{n}. \tag{17}$$

Recently, chips that can compute exponentiation in o.1 second are available. Even the chips for digital signal processing can calculate it within a second. The NEC DSP-77230 took 0.7 second. Hence the calculation of the identity-based key distribution systems do not require a big amount of time.

5. Concluding Remarks

Two types of identity-based key distribution system have been proposed for decentralized networks and centralized networks. The latter system requires only one way communication to establish a common work key. This is because it does not use the identification information of the network center. In the centralized networks, however, there have been proposed lots of key distribution systems so far. We have showed that the identity-based key distribution system does not need any public or private key-encrypting key list in the network center.

In this proposal, each system uses a common modulus n, though RSA cryptosystem cannot[6]. Moreover, the work keys are randomly determined. In the Diffie-Hellman public key distribution system, work keys between two fixed users are always a constant.

Acknowledgments

I would like to thank Masao Managaki, Katsuhiro Nakamura and Kazue Tanaka of NEC Corporation for helpful suggestions and discussions.

References

[1] DIFFIE,W., and HELLMAN,M.E.:"New directions in cryptography,"IEEE Trans., IT-22, pp.644-654, 1976.

[2] OKAMOTO,E.:"Proposal for identity-based key distribution systems," Electronics Letters, 22, pp.1283-1284, 1986.

[3] SHAMIR,A.:"Identity-based cryptosystems and signature schemes," Proc. Crypto 84, Santa Barbara, August 1984, pp.47-53.

[4] KOHNFELDER,L.:"Towards a practical public-key cryptosystem," B.S. Thesis, M.I.T., Cambridge, Mass.

[5] RIVEST,R.L., SHAMIR,A., and ADLEMAN,L.:"A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, 21, pp.120-126, 1978.

[6] SIMMONS,G.J.:"A 'weak' privacy protocol using the RSA crypto algorithm," Cryptologia, 7, pp.180-182, 1983.

SECRET DISTRIBUTION OF KEYS FOR PUBLIC–KEY SYSTEMS

Jean–Jacques Quisquater

Philips Research Laboratory Brussels
Avenue Van Becelaere, 2
B-1170 Brussels, Belgium

Extended abstract

Abstract

This paper proposes new public–key cryptosystems systems the security of which is based on the tamperfreeness of a device and the existence of secret key cryptosystems instead of the computational complexity of a *trapdoor* one–way function (RSA).

1 Introduction

This paper is an extension of ideas presented by Desmedt and Quisquater at CRYPTO '86 in [2]. In that presentation, the first identity–based cryptosystem (a nice idea proposed by Adi Shamir [3]) to protect privacy was presented. This cryptosystem is based on the existence of secret-key cryptosystems, of tamperfree devices, using an authority (a trusted center) and the key generation specific to the identity-based systems. An important open problem was set:

> *Does there exist an identity–based cryptosystem to protect privacy which security is based on tamperfree devices and computational complexity and which use* different *supersecret keys s for different users?*

We give here an explicit construction of such a cryptosystem. The involved concepts can be used to improve a signature scheme of Davies using smart cards (where there was the same problem for the unique supersecret key): Our solution allows to avoid the illegal creation of any new (false) identity in the system. A related construction permits to strongly limit the subliminal use of protocols where the exchange of random numbers is essential. We introduce in this context the concept of *certified random*.

2 Key generation for identity-based systems

The figure 1 recalls the principle of key generations for public-key cryptosystems (a) and for identity-based cryptosystems (b). The figure 2 gives a first implementation of a public-key cryptosystem where the (unique) supersecret key of the authority is s. The secret-key cryptosystems are based on the cryptographic functions E' (the "inverse" of which is D') and D''. More explanations are given in [2].

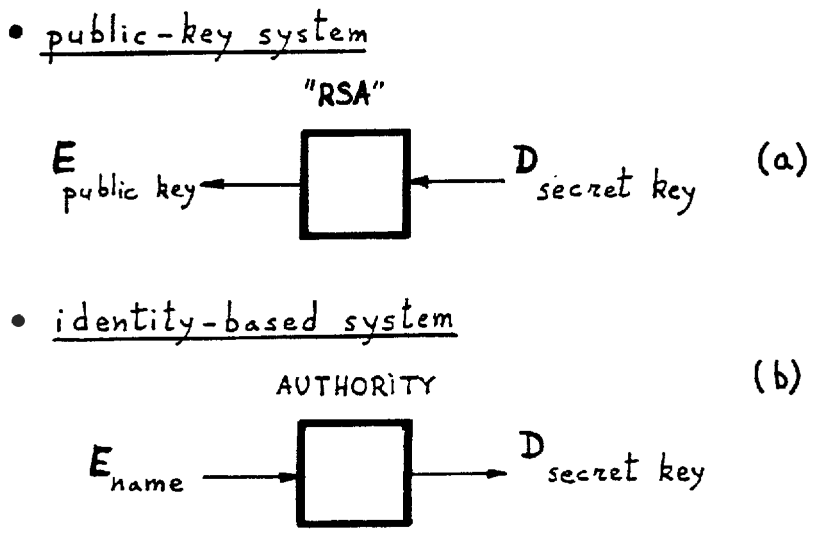


Figure 1: Key generation for public-key and identity-based systems

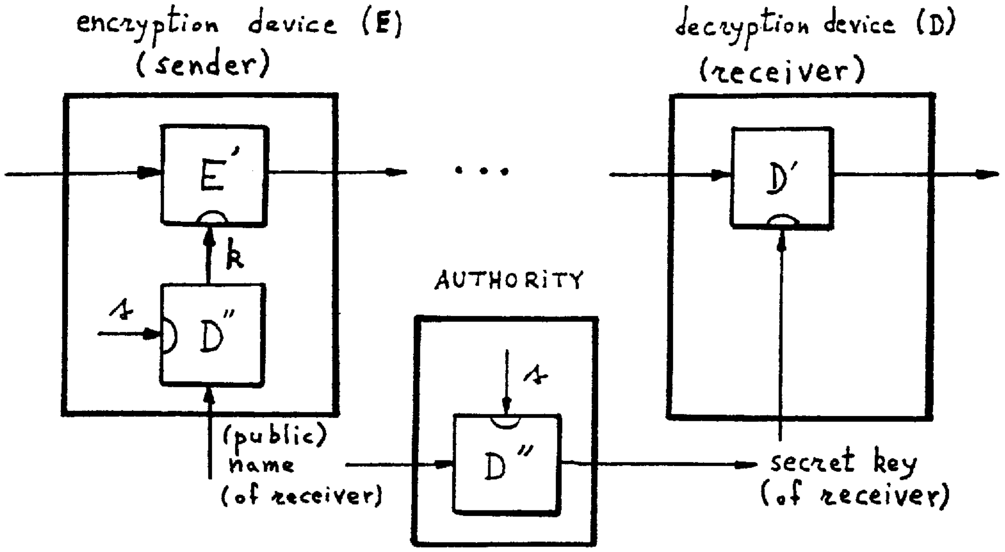


Figure 2: The first identity-based system to protect privacy

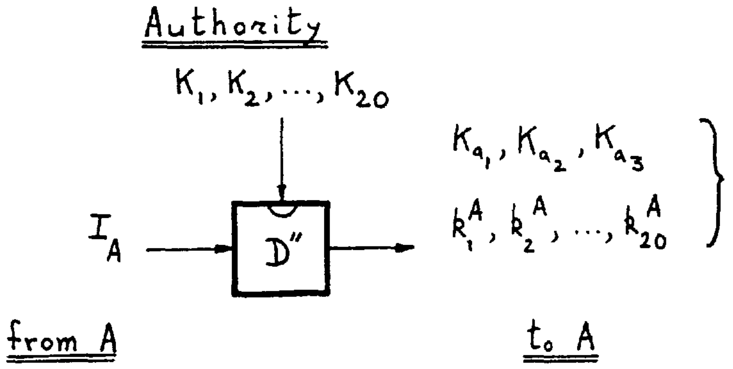

Authority
$K_1, K_2, \ldots, K_{20}$
I_A
D''
$K_{a_1}, K_{a_2}, K_{a_3}$
$k_1^A, k_2^A, \ldots, k_{20}^A$
from A
to A

Figure 3: The new key generation

3 The main idea

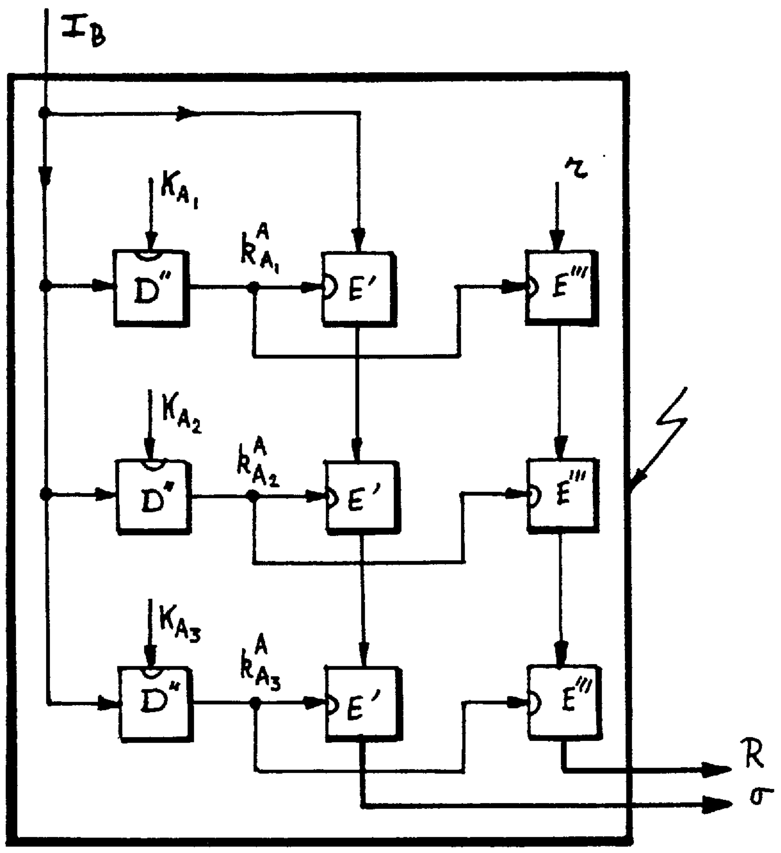
3.1 Sketch

The main idea here is that the authority has many supersecret keys (for instance, 20) but distributes only few keys to each user (for instance, 3 keys). This set of 3 keys is unique for each user (there is thus a limit for the number of users in the system, in this version) and nobody knows if he/her has one or two keys in common with a given user. The authority retains secret the distribution of keys. Each user receives also the encrypted form of his/her identity by the 20 supersecret keys. Another idea is that if one knows some *useful* information about the used keys one can avoid an exhaustive search of keys (concept of *indicator*).

Another possible and useful idea is to have two or more authorities. It will be described in the full paper.

3.2 The new key generation

The new key generation in given in figure 3. The supersecret keys of the authority are K_1, K_2, K_3, ..., K_{20}, the identity of some user A is denoted by I_A, the elements of the subset α of supersecret keys given to the user A (inside his tamperproof device) are K_{A_1}, K_{A_2} and K_{A_3} where A_1, A_2 and A_3 are ordered values chosen from $\{1, 2, 3, ..., 20\}$ by the authority in such a way that only this user has these three keys. The encrypted forms of the identity I_A by the supersecret keys are k_1^A, k_2^A, k_3^A, ..., k_{20}^A also enclosed in the tamperproof device of A.



I_B: (public) identity of B;

R: enciphered version of the session key r chosen by A;

σ: indicator of the subset α;

K_{A_1}, K_{A_2} and K_{A_3}: the 3 supersecret keys specific to the user A;

$k^A_{A_1}$, $k^A_{A_2}$, $k^A_{A_3}$: computed values, "protected" by the tamperproof device of A and also known (because it was given by the authority) by the tamperproof device of B.

Figure 4: Transmission of the session key R from A to B

3.3 The new scheme

These two combined ideas (several supersecret keys and the use of the indicator) give the scheme briefly described in figure 4. This scheme shows the transmission of a secret session key r from the user A to the user B. The sender A uses the public identity I_B of the user B as input to his tamperproof device which computes two values, transmitted to the user B;

- the enciphered session key R: the common secret key session being r;
- the indicator σ of the subset α from $\{1, 2, 3, ..., 20\}$;

Only the tamperproof device of B is able to recover the values r and α. After a maximum of $\binom{20}{3}$ computations (a practical value), the device of B recovers A_1, A_2 and A_3, using the *indicator*, that is, a value giving some indications about the used supersecret keys but useful only to him. Some collision is possible (two sets of keys give the same indicator) but it is very improbable in a well designed system and can be avoided. Some precomputations are possible to accelerate the process. Then, the device of B deciphers R using the function D''' (the inverse of E''') and the values $k^A_{A_1}$, $k^A_{A_2}$, $k^A_{A_3}$, known thanks to the indicator. The secret common key r is now transmitted.

4 Access control

Here the problem is to avoid the possibility of fraud, that is the illegal creation of certified identities (without the use of the trusted authority). We only give the main idea without detail. The context of tamperproof devices is here implicit, that is, the sensible informations are always inside such devices.

The authority has, for instance, 100 secret functions (an one-way function and 100 keys). The output of these functions can be small, one bit for instance. To each correct identity I the authority gives the 100 outputs, corresponding to these functions. Each verifier (terminal) receives only some of these functions (depending on the level of security). These functions permit to verify, with some degree of confidence, but not to create new certified identities. Again the distribution of keys by the authority is secret. So an opponent will have many problems to recover all keys in view of creating new certified identities.

5 The concept of certified random

The concept of subliminal channel was introduced by G. Simmons [4]. In fact, each protocol using random numbers has possibilities of subliminal channels. Indeed, each random number is, maybe, an encrypted message by a secret function (unknown to somebody among the parties of the protocol). How to avoid that problem? The sender of the random number has to prove to everybody the correct use of some intermediary (see the concept of warden in the paper of Y. Desmedt). That is, for instance, given a random number r, the sender computes $f(r) = R$ and sends R. Now the sender has to prove the use of f (without repeated tests to choose a "good" R). This intermediary can be a tamperproof device computing $f(r)$ and certifying this computation with the technique of the last section about access control. The verifier has some keys from the authority to verify the certificate. More details will be given in the full paper. Other techniques are possible using some concepts related to zero-knowledge protocols (with only *one* interaction).

Acknowledgement. The author is most grateful to Andrew Odlyzko and Carl Pomerance for warm encouragements.

References

[1] George Davida and Brian Matt, "*Arbitration in tamper proof systems*", *in* These proceedings, 1987.

[2] Yvo Desmedt and Jean-Jacques Quisquater, "*Public-key systems based on the difficulty of tampering (Is there a difference between DES and RSA?)*, *In* Proceedings of CRYPTO '86, Springer-Verlag, Berlin, 1987.

[3] Adi Shamir, "*Identity-based cryptosystems and signature schemes*", *in* Proceedings of CRYPTO '84, Springer-Verlag, Berlin, 1985, pp. 46-53.

[4] Gustavus J. Simmons, "*The prisoner's problem and the subliminal channel*, *in* Proceedings of CRYPTO '83, Plenum Press, New York, 1984, pp. 51–67.

An Impersonation-Proof Identity Verification Scheme*

Gustavus J. Simmons
Sandia National Laboratories
Albuquerque, NM 87185

Most schemes for the verification of personal identity are logically flawed in that they require an individual to exhibit a piece of private, i.e., secret, information such as a computer access password, a telephone credit card number, a personal identification number (PIN), etc., to prove his identity. The logical problem is that this information, once exhibited, is potentially compromised and could be used by anyone to undetectably impersonate the legitimate owner. What is needed is a protocol that will allow an individual to "prove" that he knows the secret piece of information, whose possession is equated with his identity, without revealing anything about the information itself which could aid a would-be cheater to impersonate him. Several investigators have proposed identification schemes to accomplish this [1,2,3,4] that depend on interactive-proof schemes, often referred to as zero-knowledge proofs or ping-pong protocols, in which the individual responds to a series of queries in a way that the legitimate user could, but which an impostor (probably) could not. We describe a simpler identity verification scheme which uses a public authentication channel to validate a private authentication channel belonging to the individual who wishes to prove his identity. The public and the private channels can be completely independent and can even be based on different authentication algorithms, or they can both be of the same type. This scheme also provides certified receipts for transactions whose legitimacy can later be verified by impartial arbiters who were not involved in the transaction itself.

The identity verification scheme described here presupposes the existence of a trusted issuer of validated (signed) identification credentials. This could be a government agency, a credit card center or financial institution, a military command center, a centralized computer facility, etc. The issuer first establishes a public authentication channel to which he retains the (secret) authenticating function. For simplicity, we will use the well known authentication channel based on the RSA cryptoalgorithm for both the public (issuer) and the private (user) channels, although, as mentioned earlier, authentication channels based on any other algorithm would work equally well. The issuer chooses a pair of primes p and q by the same standards used to compute a good RSA modulus, i.e., so that it is computationally infeasible for anyone to factor n, and then calculates a pair of encryption/decryption exponents, e and d such that;

* This work performed at Sandia National Laboratories supported by the U. S. Department of Energy under contract no. DE-AC04-76DP00789.

$$ed \equiv 1 \pmod{\varphi(n)} \quad .$$

n and d are made public. The issuer keeps e (and equivalently the factors p and q) secret; in fact, the security of the system against fraudulent claims of validated identity is no better than the quality of protection given to e by the issuer. The issuer also chooses a polyrandom function f that maps arbitrary strings of symbols to the range [0,n). By polyrandom we mean that f cannot be distinguished from a truly random function by any polynomially bounded computation. Many strong, single-key, cryptographic functions, such as the DES, appear to adequately approximate this condition. f is also made public by the issuer.

User i's identity is associated with an identifier, I_i, consisting of such information as his social security number, his bank account or credit card number, his military ID, etc., and which could also include physical descriptors such as digitized fingerprints, voice prints, retinal eye prints, etc., or any other useful descriptive information, as well as any limitations on the authorization conveyed in the signed identifier, such as credit limits, expiration date, levels of access, etc. Most importantly, I_i must include the public part of the user's personal authentication channel consisting of an RSA modulus n_i, $n_i > n$, and an associated decryption exponent d_i, plus, redundant information, such as message format, fixed fields of symbols common to all identifiers, I_i, etc. The issuer calculates

$$m_i = f(I_i)$$

and encrypts m_i using his secret key, e, to form the signature, s_i, for the identifier, I_i,

$$s_i \equiv m_i^e \pmod{n}$$

The issuer gives the credential (I_i, s_i) to user i. No part of the credential need be kept secret. However, the user must keep secret his private encryption exponent, e_i, corresponding to d_i. His security against impersonation is dependent on his protecting e_i, since his proof of identity in the scheme is equated to knowing e_i.

The public information is the issuer's modulus n and decryption exponent d, the polyrandom function f and a knowledge of the redundant information present in all of the I_i, which must be sufficient to prevent a forward search cryptanalytic attack [5] on the polyrandom function f. In other words, someone wishing to fraudulently validate an identity could calculate $s_j^d = m_j$ for randomly chosen signatures s_j in the hopes of obtaining a hit with f(I) for some usable I -- this is the forward search attack. By making I contain sufficient redundant information, the probability of success of this sort of attack can be made as small as desired.

When user i wishes to prove his identity to a party A, say to gain access to a restricted facility or to log on to a computer or to withdraw money from an ATM, etc., he initiates the exchange by identifying himself to A using his identification credential;

$$i \xrightarrow{(I_i, s_i):u_i} A$$

u_i is a string of symbols that describes or identifies the transaction i is requesting; u_i could be the date, the amount of the withdrawal, etc. A, who need not be a subscriber himself, i.e., he may not have an identification credential issued by the trusted issuer, replies with a string of symbols, u_A, that describe the transaction from his standpoint; terminal ID, transaction number, confirmation of withdrawal amount, etc.

$$i \xleftarrow{u_A} A$$

Both user i and A form the concatenation of u_i and u_A, $u = u_i, u_A$, and calculate the polyrandom function f(u) of the resulting string;

$$z = f(u) \quad .$$

In addition, A calculates $f(I_i) = m_i$ and s_i^d which will in all probability equal m_i modulo n if and only if the issuing authority signed I_i with s_i. Hence A accepts the credential (I_i, s_i) as valid if and only if

$$f(I_i) = s_i^d \pmod{n}.$$

At this point in the protocol, A is confident that the user identified in I_i can authenticate messages using the private authentication channel described in I_i, in other words, that user i knows e_i. In particular, user i can calculate

$$t_i = z^{e_i} \pmod{d_i}$$

using his private exponent e_i, which he communicates to A;

$$i \xrightarrow{t_i} A \quad .$$

Note that z is being used effectively as a one-time key, indeterminate to both i and A because of the polyrandom nature of f, to permit i to give A an encrypted function of z in a form that will permit A to satisfy himself that i had to know e_i without providing any information whatsoever about e_i. A knows the identity claimed by i from I_i, which he accepts as valid if and only if the following identity is satisfied:

$$t_i^{d_i} \equiv z \pmod{n_i} \qquad (1)$$

If the person seeking to be recognized as user i really is who he claims to be, i.e., if he knows e_i, then (1) will be satisfied. However, if he is not user i, so that he doesn't know e_i, then in order for him to be able to impersonate i, i.e., to cause (1) to be satisfied, he must be able to find a number x such that

$$x^{d_i} \equiv z \pmod{n_i} \quad . \qquad (2)$$

n_i or d_i are values signed by the issuer in I_i with only the authorized user knowing e_i or equivalently the factorization of n_i. z is a pseudorandom number jointly determined by user i and by A. Solving (2) without knowing e_i is equivalent to breaking the RSA cryptoalgorithm from ciphertext alone.

A keeps the 4-tuple $\left[(I_i,s_i):u,t_i\right]$ as his certified receipt for the transaction. Anyone, using only publicly available information, i.e., n, d and f, can verify that the 4-tuple satisfies (1) which validates the transaction description and verifies that it was signed, i.e., endorsed, by user i. If both communicants require a certified receipt the one-way protocol described above can be easily modified to be a two-way protocol between two parties, i and j, both of whom must possess identification credentials validated by the issuer. All actions are symmetric in this case. The exchange is of the form

$$i \quad \xrightarrow{\quad (I_i,s_i):u_i \quad} \quad j$$

$$\xleftarrow{\quad (I_j,s_j):u_j \quad}$$

$$\xrightarrow{\qquad t_i \qquad}$$

$$\xleftarrow{\qquad t_j \qquad}$$

where user i would keep the 4-tuple $\left[(I_j,s_j):u,t_j\right]$ as his certified receipt, etc.

References

1. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," Proceedings of Crypto-86, Santa Barbara, August 1986, pp. 1-13.

2. A. Shamir, "Identity-based cryptosystems and signature schemes," Proceedings of Crypto'84, Santa Barbara, August 1984, pp. 47-53.

3. O. Goldreich, S. Micali and A. Wigderson, "Proofs that yield nothing but the validity of the assertion and the methodology of cryptographic protocol design," Submitted to 27th Symposium on Foundations of Computer Science, November 1986.

4. E. Okamoto, "Proposal for identity-based key distribution systems," Electronics Letters, Vol. 22, No. 24 (Nov. 20, 1986), pp. 1283-1284.

5. G. J. Simmons and D. B. Holdridge, "Forward search as a cryptanalytic tool against a public key privacy channel," Proceedings of the IEEE Computer Society 1982 Symposium on Security and Privacy, Oakland, CA, April 26-28, 1982, pp. 117-128.

Arbitration in Tamper Proof Systems

If DES $\approx$ RSA Then What's the Difference Between True Signature and Arbitrated Signature Schemes ?

George I. Davida Brian J. Matt
Electrical Engineering and Computer Science Department
University of Wisconsin-Milwaukee
Milwaukee, WI 53201

Abstract

Given that tamperfree devices exist it is possible to construct true signature schemes that have the advantages of arbitrated signature schemes, protection against disavowing or forging messages, and lacking certain short commings. Other cryptographic protocols can also be improved. The contents of tamperfree devices cannot be examined as well as not modified.

1 Introduction

Digital signature systems not employing arbitrators are vulnerable to forging by back dating messages if secret keys are lost or stolen. One would like to avoid arbitrators, even blind arbitrators if possible. Other protocol schemes, [7] have similar difficulties. By employing "tamperfree" systems we will show that such difficulties can be avoided.

2 Digital Signatures

In true signature schemes the sender's signed messages are sent directly to the receiver. The receiver checks the validity and authenticity of the message upon receipt. The role of third parties is that of storing secret information until a dispute arises, ie *trusted third parties* or providing a *public directory*. Examples of such schemes include [9,10,11]. In arbitrated schemes "all signed messages are transmitted from S (the sender) to R (the receiver) via an arbitrator A[1], who serves as a witness."[1] For any message the arbitrator can determine the validity and authenticity of the message the sender provides and allows the receiver to be certain the message that he receives has been examined by the arbitrator.

To be an effective witness the arbitrator must:

1. Prevent a message sender from disavowing messages by leaking his secret key.
2. If the secret key is truly stolen, prevent someone from forging messages.

The system should ideally have a low operational overhead in terms of transmission costs, and computation. Both prior and especially concurrent costs should be minimized. Centralized arbitrators have to contend with message congestion and non-centralized arbitrated systems are more complex and expensive. Physical security and trustworthiness of the arbitrator are a concern especially if multiple arbitrators are necessary. This can be dealt with by employing blind arbitrators[8].

[1]It is not required that the message go directly to A. It might be sent to B first who then sends it to A.

It is issues such as these that we wish to address. First we describe a public key system based on conventional key systems due to Desmedt and Quisquater[5] then our signature system utilizing similar assumptions.

3 Tamperfree Public Key System

In "Public Key Systems Based on the Difficulty of Tampering"[5] Desmedt and Quisquater present public key cryptosystems based on conventional cryptosystems in a "tamperfree" environment. They state the following assumptions:

- Hard conventional cryptosystems exist
- It is feasible to make tamperfree devices.[5]

These "tamperfree" devices cannot be examined (to determine key values) or altered. As an example the following system was presented.

Let:	S	supersecret key
	sk	secret key (of the user)
	PK	public key (of the user)
	G	Generation algorithm for the public key
	E	Message encryption algorithm
	D	Message decryption algorithm
	E'&D'	Encryption and Decryption algorithm (1st system)
	E''&D''	Encryption and Decryption algorithm (2nd system)
	P	plain text
	C	cipher text

To generate a public key user A does the following:

$$G : E''_S(sk_A) \rightarrow PK_A$$

If user A wants to send a message to user B:

$$E : E'_{D''_S(PK_B)}(P) \rightarrow C$$

and user B decrypts by:

$$D : D'_{sk_B}(C) \rightarrow P$$

As a result of the tamperfree nature of the device user A can employ user B's public key PK_B without learning user B's secret key sk_B.

4 The Notarized System

We assume:

- The existence of a conventional cryptosystem either lacking weak keys[2] or with the existence of means of avoiding using the weak keys.
- The cryptosystems are secure, i.e. able to withstand cryptoanalytic methods.

[2]A term introduced by Davies pertaining to the weak keys of DES.[4]

- That an adequate public directory system exists.
- The existence of a cryptographic hashing function.

Each device is "tamperfree" hence the device registers cannot be examined externally. Each device contains:

1. Two key registers
2. Message counter
3. Clock
4. Encryption/decryption device
5. Cryptographic hashing function (possibly employing the encryption/decryption device)
6. An initialization flag register.

The flag register, supersecret and device secret key registers, message counter and clock employ erasable, non-volatile memory.

A device resides in one of four states, never used (pre startup), active, non-active (used and power failed) and permanently disabled (post self-destruct). Upon startup the super secret key S is set. This is the key that all devices must share for communication to be possible. The device specific secret key skd is set. The device generates a device specific public key PKd that is installed in a public directory. It is by employing this key that the device notarizes/authenticates messages. The clock is set and the message counter is initialized, probably to zero. The initialization flag is set. Once the flag is set the initialization flag itself, the message counter, and the secret keys cannot be updated. Also modification of the clock value is now restricted.

The device is now in the active state and available for cryptographic work. Keys entered during the active state are all stored in erasable, volatile memory. Should the device suffer a power failure it moves in to the non-active state.

To restore a device to the active state the clock must be reset. The time can be reset only to a value later than the time at power failure. The difference cannot be larger than some finite value or the device self destructs. The reset can only be performed by authorized personnel, this is assured by requiring that secret data be entered by the authorized person as part of the time reset process.

While in the active state each device can perform three functions, user public key generation, message encryption, and message decryption.

Let:	S	supersecret key
	sku	secret key (of the user)
	PKu	public key (of the user)
	skd	secret key (of the device)
	PKd	public key (of the device)
	G	Generation algorithm for the public key
	E	Message encryption algorithm
	D	Message decryption algorithm
	$E\&D$	Encryption and Decryption algorithm
	P	plain text
	C	cipher text
	CHF	Cryptographic Hashing Function

To generate a user public key the user A enters her secret key sku_A and selects the key generation function:

$$G : E_S(sku_A) \rightarrow PKu_A \tag{1}$$

This algorithm is also employed, using a skd_α instead of a sku_A to generate the public key of device α:

$$G : E_S(skd_\alpha) \rightarrow PKd_\alpha \tag{2}$$

For user A to encrypt a message using device α and send it to user B user A's secret key and the public key of user B are entered into the device. The message counter mc and time stamp ts are concatenated to the message and a cryptographic hash ch is made of the result. The cryptographic hashing function hashes using the user's secret key sku_A and then the device's secret key skd_α:

$$CHF_{skd_\alpha}(CHF_{sku_A}(P//mc//ts)) \rightarrow ch \tag{3}$$

This cryptographic hash provides both user A's signature and device α's notarization.

The crptographic hash is concatenated with the the message, message count and time stamp and encrypted using user B's public key.

$$E : E_{D_S(PKu_B)}(ch//P//mc//ts) \rightarrow C \tag{4}$$

The text $ch//P//mc//ts$ can be retained by user A as proof that he generated the message at the specified time.

When user B decrypts the message he enters his user secret key sku_B plus the public key of user A and device α. First the cipher text is decrypted:

$$D : D_{sku_B}(C) \rightarrow ch//P//mc//ts \tag{5}$$

Second the cryptographic hash of $P//mc//ts$ is computed and compared with ch.

$$CHF_{D_S(PKd_\alpha)}(CHF_{D_S(PKu_A)}(P//mc//ts)) \rightarrow ch' \tag{6}$$

If ch' does not match ch the message is rejected. See figure 1.[3]

5 Remarks

The scheme does not need a *Trusted third party* as do true signature schemes employing conventional cryptosystems. No separate arbitrator is employed since the devices serve as witnesses. Unlike public key based true signature schemes, the user can *properly* disavow messages by promptly informing a judge or referee of the loss. This is since the device is "tamperfree", messages cannot be forged since no one has access to a device's secret key skd and the supersecret key S, and devices cannot be coerced into back dating messages.

[3]If it is required that user B use device β to decrypt the message then user A encrypts by:

$$E : E_{D_S(PKd_\beta)}(E_{D_S(PKu_B)}(ch//P//mc//ts)) \rightarrow C \tag{7}$$

and user B decrypts by:

$$D : D_{sku_B}(D_{skd_\beta}(C)) \rightarrow ch//P//mc//ts \tag{8}$$

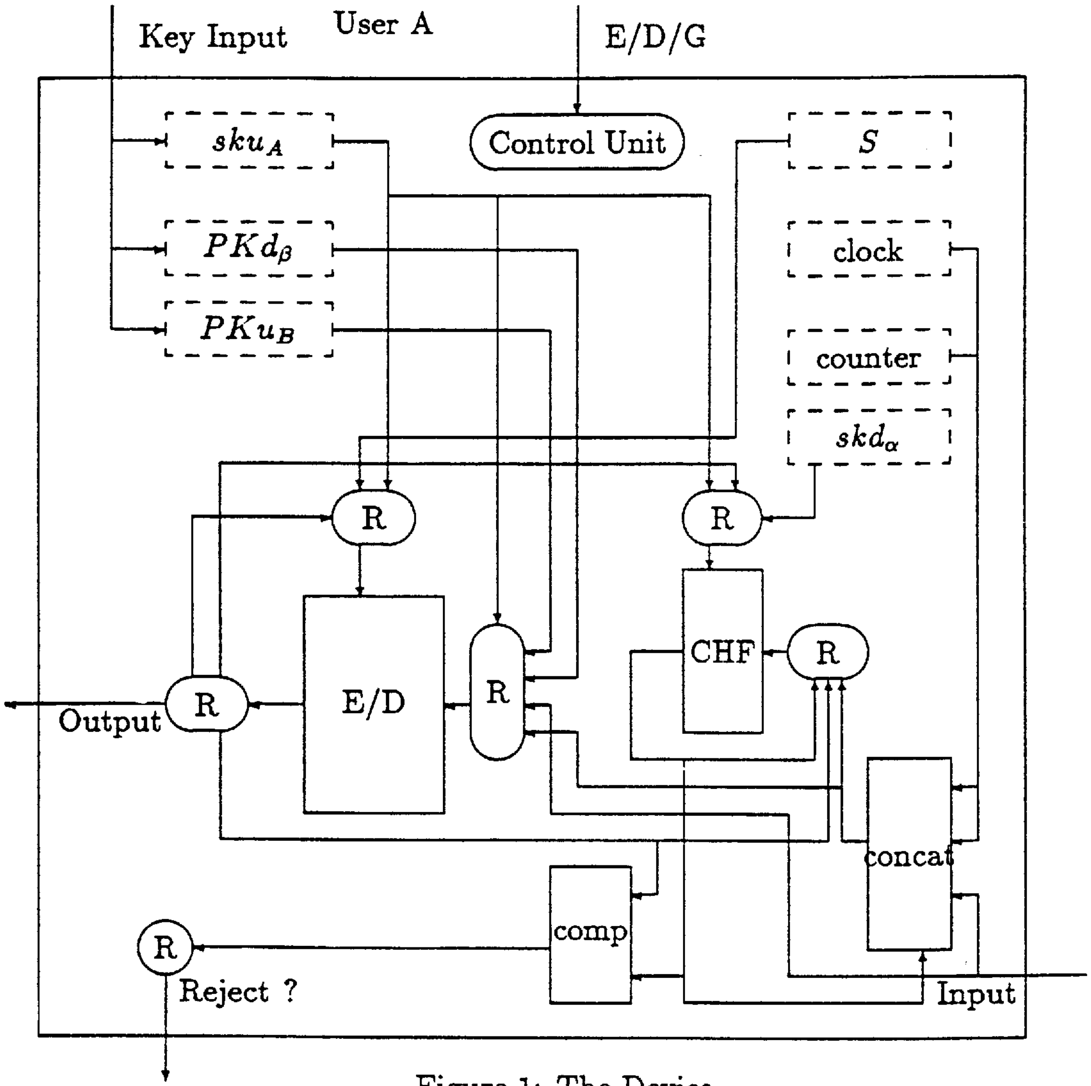


Figure 1: The Device

6 Selectively Breakable Cryptosystems

In some situations, relatively unsophisticated users engage in transactions where unaided key generation is difficult for one or both users and mutual suspicion is high. The following system was proposed:[7]

- There exists unordered pairs (A, B) of users
- There exists a *Trusted Third Party*
- The *Trusted Third Party* generates for each user pair the following:
  - A pair (R_a,R_b) of keys called the *Retrieval Keys.*
  - A pair (M_a,M_b) of keys called the *Message Keys.*

  The keys have the following properties

$$M_a : T \rightarrow C_a(T) \tag{9}$$
$$M_b : T \rightarrow C_b(T) \tag{10}$$
$$M_b : C_a(T) \rightarrow T \tag{11}$$
$$M_a : C_b(T) \rightarrow T \tag{12}$$

where T is the plain text message and C is the ciphertext.

1. M_a cannot be derived from M_b nor can M_b be derived from M_a.[4]
2. It is computationally infeasible to derive M_b or R_b from M_a and R_a and vice versa.
3. It is feasible to compute M_a and M_b from R_a and R_b.

Naturally M_a and M_b should be a good cryptosystem. In fact they should have the properties of a public key cryptosystem[6] with the addition that it should be computationally infeasible to derive either M_a or M_b from the other.

Both parties receive their message keys from the *Trusted Third Party* in such a manner that they cannot disavow the reception. The third party stores the retrieval keys in separate secure areas (physically secure and or cryptographically secure)[5] guarantees the proper delivery of the message keys. To a cryptoanalyst not in collusion with either of the parties the system appears as a conventional cryptosystem. To both A and B it appears as a public key cryptosystem with each holding the "public key".

One problem with the above scheme is that A or B could have their key stolen or claim it was stolen. Resulting as in the case of digital signatures in forged or disavowed messages. Consider a situation where the *Trusted Third party* and both A and B have devices similar to those in section number 4. Each device shares the same supersecret key and the device public key for user A's device and B's device is known to the *Trusted Third Party.* The *Trusted Third Party* merely generates a secret key and sends it to both user A and B. Both users use this key as a secret user key for communications with the other party. This communication is through each user's device and the messages are stamped as in section 4. This solution does not require any additional hardware for user's A and B.

[4]Actually this follows from condition two.

[5]If the *Trusted Third Party* cannot be trusted so far a key sharing scheme can be employed. [2,12,3]

7 Acknowledgement

The authors wish to Professor Yvo Desmedt for several helpful suggestions.

References

[1] Selim G. Akl. Digital signatures: a tutorial survey. *IEEE Computer*, 16(2):15–24, 1983.

[2] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979 NCC*, pages 313–317, 1979.

[3] George I. Davida, Richard A. DeMillo, and Richard J. Lipton. Protecting shared cryptographic keys. In *Proc. 1980 Symp. on Security and Privacy*, pages 100–102, IEEE, April 1980.

[4] Donald W. Davies. Some regular properties of the 'data encryption standard'. In *Advances in Cryptology: Proceedings of Crypto82*, pages 89–96, 1983. Actually presented at Crypto-81.

[5] Yvo Desmedt and Jean-Jacques Quisquater. Public key system based on the difficulty of tampering (is there a diffference between des and rsa?). In *CRYPTO'86 Abstracts and Papers*, pages 15-1 - 15-3, 1986.

[6] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. on Inform. Theory*, 22(6):644–654, 1976.

[7] Brian J. Matt. *Selectively Breakable Cryptosystems and Personal Keys*. Master's thesis, University of Wisconsin Milwaukee, May 1981.

[8] Henk Meijer and Selim Akl. Digital signature schemes. *Cryptologia*, 6(4):329–338, October 1982.

[9] R. C. Merkle. Protocols for public key cryptosystems. In *Proc. 1980 Symp. on Security and Privacy*, pages 122–134, IEEE, April 1980.

[10] M. Rabin. Digitalized signatures. In R. A. DeMillo et al., editors, *Foundations of Secure Computation*, pages 155–166, 1978.

[11] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptograms. *Communications of the ACM*, 21(2), Feburary 1978.

[12] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

EFFICIENT DIGITAL PUBLIC-KEY SIGNATURES WITH SHADOW

Louis Guillou * & Jean-Jacques Quisquater **

(*) CCETT, Rue du Clos-Courtel, B.P. 59, F-35510 Cesson-Sévigné, France.

(**) Philips Research Laboratory Brussels, Avenue Van Becelaere, 2, B-1170 Brussels, Belgium.

Abstract

This paper describes a strictly deterministic digital signature scheme using public-key cryptosystems. This scheme is in discussion inside a working group of ISO on signature schemes (TC97/SC20/WG2). A working draft has been written and accepted recently (with formal modifications to be added). By this presentation we hope to receive useful remarks for improving this scheme. This scheme is the fastest known scheme for the *verification* of signature (only one square plus some very easy operations).

We introduce the notion of signatures with *shadow* – that is, when the signature is mathematically attached together the message to sign – and with *imprint* – that is, where the message and the signature are two separate entities.

This scheme is RSA-based and, in some aspects, is a variant of Rabin-Williams schemes. However, if the scheme is correctly implemented, it is immune against the attacks by chosen messages. The technique is well-known: You introduce a redundancy in the message before you sign with your secret function. This trick permits to the secret function to be defined nearly nowhere: The multiplicative attack of Davida, for instance, is then without effect.

The original part of the paper is an effective and efficient proposition for the function of redundancy and its uses and a structured way to sign and to verify with a weak dependence from the functions. If you use the RSA or one of its variants, you are in a context where each known optimisation is possible.

The function of redundancy is defined in such a way that the system resists to any known attack; moreover the redundancy is used, if necessary, to identify the correct message from the possible ones when you verify the signature . Indeed, in the scheme of Williams, the operation of signature needs a square root: The message thus needs to be a quadratic residue. A useful result is that from the set $\{a, -a, a/2, -a/2\}$, there is one and only one quadratic residue . So, in fact, you can detect the unique quadratic residue from this set then you extract the square root of this one. The redundancy is also used to indicate the length of the signed message (if the message is short) or of the hashed value of the message.

Many useful details will be given in the full paper.

Security-Related Comments Regarding McEliece's Public-Key Cryptosystem

Carlisle M. Adams
Henk Meijer

Department of Computing and Information Science
Queen's University, Kingston, Canada

August 1986

Abstract

The optimal values for the parameters of the McEliece public key cryptosystem are computed. Using these values improves the cryptanalytic complexity of the system and decreases its data expansion. Secondly it is shown that the likelihood of the existence of more than one trapdoor in the system is very small.

Keywords: Public key cryptosystems, Goppa codes.

1. Introduction

McEliece [1978] has introduced a public-key cryptosystem which is based on algebraic coding theory. In this system, the receiver first constructs an easily-solvable linear error-correcting Goppa code C with generator matrix G and then transforms this matrix into G', a generator matrix for a seemingly difficult-to-solve linear code C'. The matrix G' is the public key of this system -- a message is encrypted by multiplying it with G' and adding errors to the resulting codeword. The legitimate receiver can recover the message by using a decoding algorithm for the original code C.

In the first part of this paper we compute the optimal number of errors that should be introduced in the encryption algorithm; in the second part we comment on the likelihood of finding transformations that will map the code C' into the code C, or into some other easily-solvable Goppa code. The results indicate that introducing the optimal number of errors yields very high security for this cryptosystem and that there is, with high probability, only one transformation from C' into an easily-solvable Goppa code (this is the transformation known to the receiver).

We will begin this paper with a short description of McEliece's system. We continue in Sections 3 and 4 with an analysis of two attacks which may be considered by an intended eavesdropper and close, in the final section, with some concluding remarks on our results.

This work was partially supported by Grant #A0282 from the Natural Sciences and Engineering Research Council.

2. McEliece's System

McEliece's public-key cryptosystem can be briefly described as follows:

1) The receiver constructs an easily-solvable, binary, error-correcting Goppa code C which has a $(k \times n)$ generator matrix G and an error-correcting capability of t errors (note that G is necessarily of full rank).

2) The matrix G is transformed by

$$G' = SGP \tag{1}$$

where S is a $(k \times k)$ invertible scrambling matrix and P is an $(n \times n)$ permutation matrix. The $(k \times n)$ matrix G' is then a generator matrix for an apparently arbitrary linear code C' (i.e. one for which a fast algorithm for correcting errors is not known).

3) G' is published as the encryption key; the sender encrypts a k-bit message vector $\underline{m}$ into n-bit ciphertext vector $\underline{c}$ by

$$\underline{c} = \underline{m}\, G' \oplus \underline{e}, \tag{2}$$

where $\underline{e}$ is an n-bit error vector of weight t chosen by the sender.

4) The receiver, knowing that

$$\underline{c} = \underline{m}\, G' \oplus \underline{e}$$
$$= \underline{m}\, SGP \oplus \underline{e}$$

computes

$$\underline{c}\, P^{-1} = (\underline{m}\, S)\, G \oplus \underline{e}\, P^{-1}$$

and uses a decoding algorithm for the original code C to remove the error vector $\underline{e}\, P^{-1}$ and recover the vector $\underline{m}$ S. The sender's message is then easily found by

$$\underline{m} = (\underline{m}S)\, S^{-1}.$$

The private key for this system, therefore, consists of the three matrices G, S, and P.

This paper is mainly concerned with equations (1) and (2) above. We begin by calculating the optimal weight of the error vector $\underline{e}$ in equation (2), where a weight is "optimal" if it yields maximum security for this system. We then go on to consider equation (1). From (1) we have

$$G = S^{-1} G' P^{-1}. \tag{3}$$

We will compute the expected number of matrices S_i and P_i such that the code C_i with generator matrix

$$G_i = S_i G' P_i$$

is an easily-solvable Goppa code.

3. Parameters k and t.

As noted in [Adams 1985] and [McEliece 1978], there are several ways of attacking McEliece's cryptosystem. Of the known attacks, the one which follows has the lowest complexity. We will show how a suitable choice of parameters k and t will maximize this complexity and thus strengthen the algorithm against this attack.

Recall equation (2):

$$\underline{c} = \underline{m}\, G' \oplus \underline{e}.$$

Since $\underline{m}$ is a k-bit vector, we can reduce this to

$$\underline{c}_k = \underline{m}\, G'_k \oplus \underline{e}_k,$$

where $\underline{c}_k$ denotes any k components of $\underline{c}$ (ie. $\underline{c}_k = c_{i_1}, c_{i_2}, \ldots, c_{i_k}$), $\underline{e}_k$ denotes the corresponding k components of $\underline{e}$, and G'_k is the square matrix consisting of columns $i_1, i_2, \ldots, i_k$ of G'. Thus we have

$$\underline{c}_k \oplus \underline{e}_k = \underline{m}\, G'_k$$

or, if G'_k is invertible,

$$(\underline{c}_k \oplus \underline{e}_k)\,(G'_k)^{-1} = \underline{m}. \qquad (4)$$

Note that if the k components of $\underline{e}_k$ are all zero, (4) reduces to

$$\underline{c}_k\,(G'_k)^{-1} = \underline{m}$$

and an enemy can recover the sender's message without decoding (since $\underline{c}$ and G' are known).

The work factor for this attack can be calculated as follows. The error vector $\underline{e}$ is an n-bit vector with t ones and n-t zeros. Therefore, the probability of choosing (without replacement) k zero components from $\underline{e}$ is

$$p = \binom{n-t}{k} \Big/ \binom{n}{k}.$$

Note that the enemy must, on average, make 1/p attempts before being successful and, for each attempt, must invert the (k × k) submatrix G_k. Assuming that matrix inversion requires k^a steps (see, for example, [Bunch, Hopcroft 1974] and [Pan 1978]), this gives a total expected work factor for this attack of

$$w = k^a \binom{n}{k} \Big/ \binom{n-t}{k} \text{ steps.} \qquad (5)$$

From [Berlekamp 1973] or [McEliece 1977] it can be seen that for $n = 2^i$, n, k, and t are related by

$$k = 2^i - it = n - it.$$

Therefore, for n = 1024 (as suggested in [McEliece 1978]), we have k = 1024 - 10t. It can easily be shown by exhaustive search that for values of a between 2 and 3 the value of (5) is maximal for t = 37. For a = 3, the value of (5) is approximately $2^{84.1}$ for t = 37, while for t = 50 (the value proposed in [McEliece 1978]) (5) has the value $2^{80.7}$.

Moreover the lower value of t increases the value of k from 524 to 654 and therefore reduces the data expansion of the cryptosystem.

4. Trapdoors

Brickell [1985] has shown that iterated knapsack cryptosystems (proposed in [Merkle, Hellman 1978]) can be broken. The idea of the proof is that the public (difficult) knapsack can be transformed by the enemy into one of several easy knapsacks; finding the receiver's original easy knapsack is not necessary. We can examine McEliece's system in this light by estimating the likelihood of there being several transformations from the public key G' into an easily-solvable decoding problem.

To do this we define an equivalence relation R on the set of binary $(k \times n)$ matrices of full rank as follows:

A R B if and only if there exists a $(k \times k)$ invertible matrix S and an $(n \times n)$ permutation matrix P such that $A = S\,B\,P$.

If we call [A] the equivalence class induced by R containing A then it is clear that the private matrix G of McEliece's system is in the equivalence class [G'] of the public matrix G'. However, if there are other Goppa code generator matrices in the equivalence class [G'], a Brickell-like attack on this cryptosystem may be feasible.

If we assume that Goppa code generator matrices are evenly distributed over the set of all $(k \times n)$ matrices of full rank, we can calculate the expected number EXP of Goppa code generator matrices in an equivalence class of R by

$$\mathrm{EXP} = \#G / \#C \qquad (6)$$

where #G is the number of Goppa code generator matrices for a given n and k and #C is the number of equivalence classes of R. To our knowledge, the above assumption has never been mentioned in the open literature (likely due to the fact that Goppa code generator matrices are not, in general, recognizable as such).

The values #G and #C have been computed in [Adams 1985]. For Goppa codes with error-correcting capability $t = 50$ and dimension $k = 524$ (the parameters suggested in [McEliece 1978]), #G is less than or equal to the number of irreducible polynomials of degree 50 over $GF(2^{10})$ (for $n = 2^{10} = 1024$) and #C is roughly equal to the total number of binary (524×1024) matrices of full rank divided by the average size of an equivalence class. Substituting the calculated values into (6) we have

$$\mathrm{EXP} < 2^{504}/2^{500000} << 1. \qquad (7)$$

It can be shown that for $t = 37$ and $k = 654$ (from the previous section) the value of EXP is even smaller.

Given the above assumption, then, equation (7) shows that we expect that an arbitrary equivalence class of R does not contain a generator matrix for a Goppa code. From the construction of the cryptosystem, however, we know that the equivalence class of G' contains the receiver's private matrix G; therefore, we conclude from (7) that G is the only Goppa code generator matrix in [G']. Thus, the only transformation from G' to an easy generator matrix is the original transformation (3) chosen by the receiver and a Brickell-like attack against this system will be unsuccessful.

5. Conclusions

We conclude that McEliece's public-key cryptosystem appears to be fairly secure. We have shown that the lowest complexity cryptanalytic attack yet proposed has a work factor of roughly 2^{84} steps -- this is significantly higher than that of DES and compares very favourably with that of the RSA system. Furthermore, it seems that an attack similar in nature to Brickell's attack on the Knapsack cryptosystem will be unsuccessful.

References

Adams, C.M. (1985), Examination and Analysis of McEliece's Public-Key Cryptosystem, M.Sc. Thesis, Department of Computing and Information Science, Queen's University, Kingston.

Berlekamp, E.R. (1973), Goppa Codes, IEEE. Transactions on Information Theory, Vol. IT-19 #5 (Sept.).

Brickell, E.F. (1985), Breaking Iterated Knapsacks, Advances in Cryptology: Proceedings of Crypto 84, Blakley, G.R., Chaum, D. (Editors), Springer-Verlag, Berlin.

Bunch, J., Hopcroft, J.E. (1974), Triangular Factorization and Inversion by Fast Matrix Multiplication, Mathematics of Computation, Vol. 28; 125.

McEliece, R.J. (1977), The Theory of Information and Coding (Volume 3 of the Encyclopedia of Mathematics and its Applications), Addison-Wesley, Reading, Mass.

McEliece, R.J. (1978), A Public-Key Cryptosystem Based on Algebraic Coding Theory, DSN Progress Report (Jan, Feb), Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Calif.

Merkle, R., Hellman, M. (1978) Hiding Information and Signatures in Trapdoor Knapsacks, IEEE. Transactions on Information Theory, Vol. IT-24 #5 (Sept.)

Pan, V. (1978), Strassen's Algorithm is not Optimal, the 19th Annual Symposium on the Foundations of Computer Science.

Components and Cycles of a Random Function[*]

J. M. DeLaurentis
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

This investigation examines the average distribution of the components and cycles of a random function. Here we refer to the mappings from a finite set of, say, n elements into itself; denoted by Γ_n. Suppose the elements of Γ_n are assigned equal probability, i.e. $P(\gamma) = n^{-n}$, $\gamma \in \Gamma_n$. The directed graph that is naturally associated with γ consists of several components, each with a unique cycle. Define $X_n(s,t)(\gamma)$ as the number of components in γ containing at least the fraction s of the total number of nodes, with the size of each component's cycle not exceeding $tn^{1/2}$. We show that the expected value of $X_n(s,t)$ can be approximated by the double integral

$$EX_n(s,t) \approx \int_t^1 \int_0^s \frac{1}{\sqrt{2\pi}} \frac{\exp[-y^2/(2x)]}{\sqrt{x^3(1-x)}} \, dydx \quad .$$

The average number of components of a given size with cycles of a specified length approximately equals the volume under the graph of the integrand. This expression can be used to estimate the probability that a function has a component which contains a significant percentage of the total number of nodes and yet its cycle is relatively small.

Introduction

Random functions often arise as a model for the pseudo-random functions generated by a cryptosystem. Typically, the experiments performed on the latter are concerned with the

[*]This work performed at Sandia National Laboratories supported by the U.S. Dept. of Energy under contract No. DE-AC04-76DP00789.

size of its components and the length of the corresponding cycles. It is natural then to address similar problems for random mappings. This paper analyses the expected number components of a given size that contain cycles of a specified length. The asymptotic expression of this average value for random mappings provides some insight into the behavior of pseudo-random functions.

To better understand Hellman's time-memory cryptanalytic scheme [1], Hellman and Reyneri [2] estimated the expected size of the largest component of a random function. They compare this value with the average size of the largest component of the pseudo-random functions generated by the Data Encryption Standard (DES). In this case they considered mappings $f(\cdot)$ from the key-space into itself. Specifically, the functional value $f(k)$ was defined by applying the DES operation $S_k(\cdot)$ to a fixed plaintext block P_0 and then reducing the 64-bit block $S_k(P_0)$ to 56-bits through a reduction operation $R(\cdot)$,

$$f(k) = R(S_k(P_0)) \ .$$

Choosing a different plaintext block defines a new function. They found their statistical tests to be in close agreement with the expected outcome for random maps.

More recently, at Crypto '86, G. J. Simmons presented a study by Quisquater [3] in which the cycling experiments involved DES functions similar to those described above. In his investigation Quisquater found a function with a relatively large component ($\approx$ 3% of the total number of nodes) that contained a relatively small cycle (cycle size $\approx 2^{16}$). As we will see, the probability of such an event for random functions is $\approx 10^{-3}$.

A variety of different functions have been introduced to analyze cryptosystems. To examine the closure properties of DES, Kaliski et al. [4] defined a set of mappings from the cipher-space into itself. In contrast to the preceding example, they applied a pseudo-random function $g(\cdot)$ to the cipher x to obtain a key $k = g(x)$. In turn, this key was used in the DES operation to produce

$$f(x) = S_k(x) = S_{g(x)}(x) \ .$$

Again, these studies detected no statistical anomalies. Assuming that random functions are a reasonable model for the maps in question, the methods developed in this paper are applicable.

Beyond these practical motivations the study of random functions is of some intrinsic combinatorial interest. Examples of probability distributions related to random functions are presented in [5], [6], [7], and [8]. A relationship between branching processes and random maps is discussed in [9] and [10]. For a survey of results, see [11].

In the following section we introduce the necessary definitions and notation. First we consider the average number of components of size k containing a cycle of length ℓ (a (k,ℓ)-component). Next we analyze the expected number of such components when k and ℓ are allowed to range over an entire region. Finally we estimate the probability of discovering a function that has a component which contains a significant percentage of the total number of nodes and such that its cycle is relatively small.

<u>Components and Cycles</u>

The set of mappings from an n element set into itself endowed with the uniform distribution is called the set of random functions or random mappings and is denoted $\Gamma_n, (P(\gamma)=n^{-n}, \gamma\in\Gamma_n)$. The directed graph naturally associated with each function γ is the graph with a vertex for each element of the domain and a directed edge from vertex i to vertex j if and only if $\gamma(i) = j$. The components of such a graph consist of a cycle with trees attached to its nodes (cyclic points). (In the following we sometimes write that γ has a certain property when in fact it is the associated graph that has this property.) We are interested in estimating the average number of components of a given size and with a specified cycle length.

First we define the random variable $Y_n(k,\ell)(\gamma)$ that counts the number of (k,ℓ)-components in γ (a (k,ℓ)-component has k nodes and ℓ cyclic points). Next we introduce the function

$$f(x,y) = \frac{1}{\sqrt{2\pi}} \frac{\exp[-y^2/2x]}{\sqrt{x^3(1-x)}} .$$

The following lemma, whose proof is postponed until the end of this section, provides an estimate for the average value of $Y_n(k,\ell)$.

Lemma

For $Y_n(k,\ell)$ defined as above with $1 \le \ell \le k/2$ and $k \le n - n^{1/3}$ we have

$$E\, Y_n(k,\ell) = f(x_k, y_\ell) n^{-3/2} \, (1 + R_n(k,\ell)) \, , \tag{1}$$

where $x_k = k/n$, $y_\ell = \ell/\sqrt{n}$ and $R_n(k,\ell) = O(\ell^3/k^2 + k^{-1} + n^{-1/3})$. For $n - n^{1/3} < k \le n$ the left hand side is $O\left[f(x_{k-1}, y_\ell) n^{-3/2}\right]$.

The main features of this result are best explained by means of the graph of f(x,y) (see figure 1).

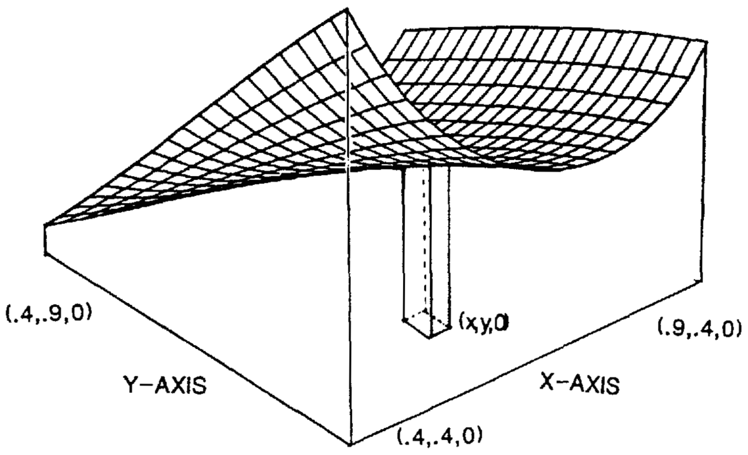


fig 1 GRAPH OF F (X, Y)

From the graph we see that the expected number of (k,ℓ)-components approximately equals the volume of the parallelepiped with height $f(x_k,y_\ell)$ whose basis is the rectangle of area $n^{-3/2}$ centered at (x_k,y_ℓ).

Except for k extremely close to 0 or n the sum over a range of these average values approximately equals the volume under the graph of f over a given region. For example, let $X_n(s,t)$ represent the number of (k,ℓ)-components with $sn \le k \le n$ $(0<s<1)$ and $1 \le \ell \le tn^{1/2}$, that is

$$X_n(s,t) = \sum_{sn\le k\le n} \sum_{1\le\ell\le tn^{1/2}} Y_n(k,\ell) \ . \tag{2}$$

Excluding the terms $n-n^{1/3} < k \le n$, the expected value of the expression on the right is asymptotically equivalent to the integral of $f(x,y)$ over $s \le x \le 1 - n^{-2/3}$, $0 \le y \le t$. The sum involving the excluded terms is measured by the integral of $\min\{t,1\} \left[x^3(1-x)\right]^{-1/2}$ over the interval $1 - n^{-2/3} \le x \le 1$, and the latter is $O(\min\{t,1\}n^{-1/3})$. It follows from the lemma that for s fixed, $0 < s < 1$, and n sufficiently large, we have:

<u>Theorem 1</u>

The average number of (k,ℓ)-components in the region $sn \le k \le n$, $1 \le \ell \le tn^{1/2}$ is approximately

$$EX_n(s,t) = \int_s^1 \int_0^t f(x,y)dydx\left[1+R_n(s,t)\right] \ , \tag{3}$$

where $R_n(s,t) = O(t^3s^{-2}n^{-1/2}+s^{-1}n^{-1}+n^{-1/3})$.

Note: The error made by replacing the sum with the integral has been included in the remainder term. Also, notice that the x-coordinate refers to the component's size and the y-coordinate indicates cycle length (see figure 2).

Although (3) is an asymptotic expression for an expected value, it can be used to estimate probabilities. As an example, we consider the probability of finding a function that has a component which contains a significant percentage of the total number of nodes and yet its cycle is relatively small. That is, we want to estimate the probability that a random function has a (k,ℓ)-component in which k is a significant fraction of the total number of nodes and yet ℓ is relatively small.

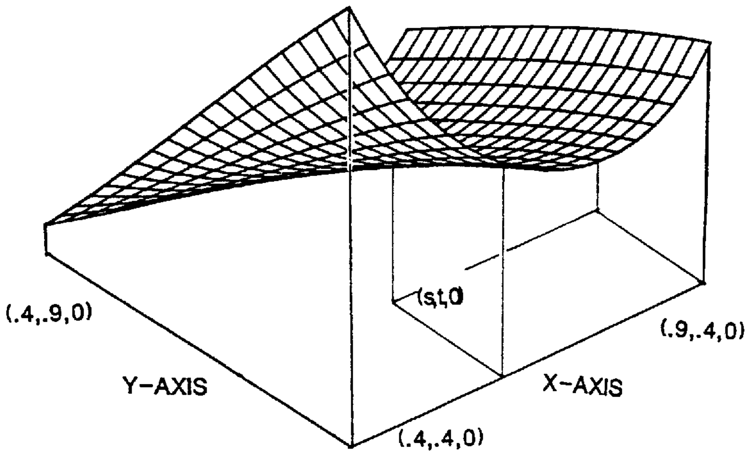


fig 2 GRAPH OF F (X, Y)

To make these terms precise, we consider a cycle of order $n^{1/2-\alpha}$, $0 < \alpha < 1/2$, as small (since the total number of cyclic points is on average $n^{1/2}$). A component is considered large if it contains at least the fraction s of the total number of nodes, $0 < s < 1$. The number of such components is given by $X_n(s,t)$ where $t = n^{-\alpha}$. Using this notation the problem is to estimate the probability that $X_n(s,t)$ is at least one.

The main idea is that if a function possesses a relatively large component with a small cycle, then it is most likely the only such component. It follows that

$$P(X_n(s,t) \geq 1) \approx EX_n(s,t) \ . \tag{4}$$

More precisely, by Bonferroni's inequality [12], we have

$$EX_n - E\left[X_n(X_n-1)\right] \leq P(X_n \geq 1) \leq EX_n .$$

It can be shown (see the appendix) that $E\left[X_n(X_n-1)\right]$ is $O(s^{-1}n^{-2\alpha})$. Thus we need only estimate the mean of $X_n(s,t)$.

Omitting, for the present, the error terms in Theorem 1, we obtain

$$EX_n(s,t) \approx (2\pi)^{-1/2} \int_s^1 x^{-3/2}(1-x)^{-1/2} \int_0^t e^{-y^2/2x} dydx \ , \tag{5}$$

recall that $t = n^{-\alpha}$. For small t the innermost integrand approxmiately equals one. That is, replacing the exponential in (5) with $1 + O(y^2/s)$ yields

$$EX_n(s,t) \approx (2\pi)^{-1/2} n^{-\alpha} \int_s^1 x^{-3/2}(1-x)^{-1/2}dx\left[1+O(s^{-1}n^{-2\alpha})\right] \ . \tag{6}$$

Fortunately, the integral in (5) can be evaluated explicitly as

$$c(s) \equiv \left[\frac{2(1-s)}{\pi s}\right]^{1/2} = (2\pi)^{-1/2} \int_s^1 x^{-3/2}(1-x)^{-1/2}dx \ . \tag{7}$$

Using the estimate for the second factorial moment and including the remainder terms given in (3) leads to the conclusion

<u>Theorem 2</u>

The probability of finding at least one (k,ℓ)-component with $sn \le k \le n$, $1 \le \ell \le tn^{1/2}$ where $0 < s < 1$ and $t = n^{-\alpha}$ is given by

$$P(X_n(s,t) \ge 1) = c(s)n^{-\alpha}\{1+O[r_n(s)]\} \ , \tag{8}$$

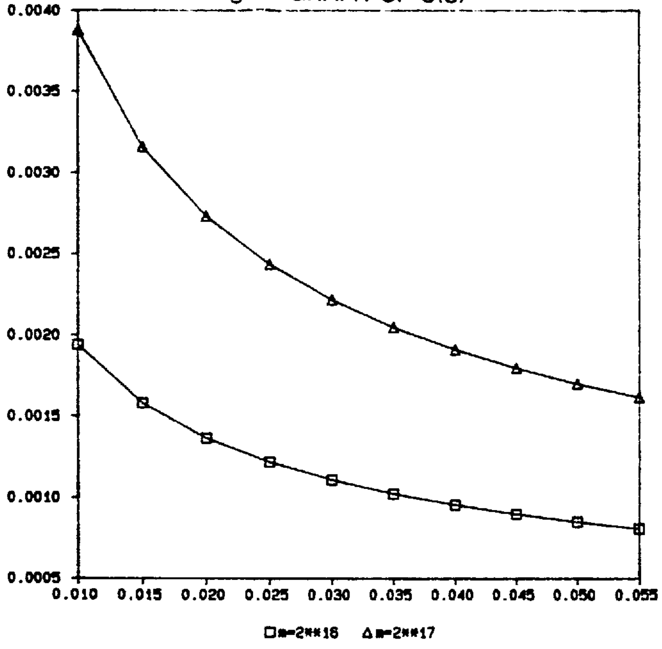
here $r_n(s) = s^{-2}n^{-1/2-3\alpha} + s^{-1/2}n^{-\alpha} + n^{-1/3}$.

Consider the example presented in the introduction [3]. In this case $n = 2^{56}$, $tn^{1/2} = 2^{16}$ (i.e., $t = n^{-3/14} = 2^{-12}$ and $\alpha = 3/14$), and $s = 0.03$. It follows that

$$P(X_n(s,t) \ge 1) \approx c(0.03)2^{-12} \approx 10^{-3} \ ,$$

with $r_n(0.03) \le 2x10^{-3}$ (see figure 3). The key observation is that even if a DES cycling experiment occasionally produces a relatively large component with a small cycle, this does not necessarily imply a statistical irregularity in DES.

fig 3 GRAPH OF C(S)



Note: Here $m = tn^{1/2} = n^{1/2-\alpha}$ is the maximal cycle length.

We turn now to the proof of the lemma. Let $C(k,\ell)$ denote the number of connected mappings on k nodes with ℓ cyclic points. It is known [11] that

$$C(k,\ell) = (\ell-1)! \begin{bmatrix} k-1 \\ \ell-1 \end{bmatrix} k^{k-\ell} \quad . \tag{9}$$

Using Stirling's formula [12]

$$k! = \sqrt{2\pi k} \left(\frac{k}{e}\right)^k \left[1 + O(\tfrac{1}{k})\right] \quad , \tag{10}$$

we compute

$$C(k,\ell) = k^{k-1}e^{-\ell^2/2k}\ [1 + O(\ell^3/k^2)] \quad , \tag{11}$$

for $1 \leq \ell \leq k/2$.

The main idea in the proof involves writing $Y_n(k,\ell)$ as the sum of identically distributed random variables. We define $\epsilon_i = 1$ if the i-th node belongs to a (k,ℓ)-component, otherwise set $\epsilon_i = 0$.

It follows that

$$Y_n(k,\ell) = \frac{1}{k} \sum_{1 \leq i \leq n} \epsilon_i \text{ and } E(Y_n(k,\ell)) = \frac{n}{k} P(\epsilon_i{=}1) \quad . \tag{12}$$

The probability that the i-th node belongs to a (k,ℓ)-component is given by

$$P(\epsilon_i{=}1) = \binom{n-1}{k-1} C(k,\ell)\ (n-k)^{n-k}/n^n \quad . \tag{13}$$

The first term is the number of ways to select the other members of the (k,ℓ)-component; the second term is the number of connected mappings consisting of k nodes and ℓ cyclic points; the third term is the number of functions on the n-k remaining elements; and the last term is the total number of mappings on an n element set. Combining (11) - (13) yields the expression

$$E\left[Y_n(k,\ell)\right] = \binom{n}{k}\ (n-k)^{n-k}k^{k-1}e^{-\ell^2/2k}n^{-n}[1{+}O(\ell^3/k^2)] \quad . \tag{14}$$

Applying Stirling's formula (10) to the first term in (14) and simplfying leads to the desired result

$$E\ [Y_n(k,\ell)] = f(x_k,y_\ell)n^{-3/2}[1{+}O(\ell^3/k^2{+}k^{-1}{+}n^{-1/3})] \quad , \tag{15}$$

for $1 \le \ell \le k/2$, $k \le n-n^{1/3}$, $x_k = k/n$ and $y_\ell = \ell/\sqrt{n}$. The proof of the last statement in the lemma is similar. The key step in these arguments is the introduction of the auxiliary random variables ϵ_i.

Summary

To better understand the structure of random functions we have examined the average distribution of its components and cycles. The relationship between a component's size and its cycle length is best illustrated by the graph of f(x,y). The volume under the graph and over a specified region represents the expected number of components in a given range with cycle lengths belonging to a prescribed interval. In turn this mean value is used to estimate the probability of discovering a function containing a relatively large component with a small cycle.

Appendix

The derivation of the asymptotic expression for the second factorial moment $E[X_n(X_n-1)]$ is similar to the development for the mean of X_n. First, we may assume that $s \le 1/2$ since $X_n(s,t)(X_n(s,t)-1) = 0$ if $s > 1/2$. Expanding the product yields

$$X_n(s,t)(X_n(s,t)-1) = \sum_{\substack{sn\le k,k' \\ k'+k\le n}} \sum_{1\le \ell,\ell' \le tn^{1/2}} Y_n(k,\ell)[Y_n(k',\ell')-\delta] , \tag{16}$$

where $\delta = 1$ if $k = k'$ and $\ell = \ell'$; otherwise $\delta = 0$. So the problem is reduced to estimating the mean value of the terms in the sum.

As in the proof of the lemma, the main idea is to represent these terms as the sum of identically distributed random variables. Fix (k,ℓ) and (k',ℓ'); set $\epsilon_{ij} = 1$ if i,j belong to different (k,ℓ), (k',ℓ')-components, respectively; otherwise, let $\epsilon_{ij} = 0$. A straightforward calculation shows that

$$Y_n(k,\ell)[Y_n(k',\ell')-\delta] = \frac{1}{kk'} \sum_{i\neq j} \epsilon_{ij} \quad . \tag{17}$$

The average value of the right-hand side of (17) is given by

$$\frac{n(n-1)}{kk'} P(\epsilon_{ij}=1) \tag{18}$$

$$= \frac{n!}{k!k'!(n-k-k')!} C(k,\ell)C(k',\ell')(n-k-k')^{n-k-k'} n^{-n} \quad .$$

Here we have used arguments similar to the ones employed in the derivation of (13). As before we apply Stirling's formula (10) and expression (11) to obtain the estimate

$$E\,\{Y_n(k,\ell)[Y_n(k',\ell')-\delta]\} = O[g(x_{k-1},x_{k'-1})n^{-3}] \quad , \tag{19}$$

with

$$g(x,x') = x^{-3/2}x'^{-3/2}(1-x-x')^{-1/2}$$

and $x_k = k/n$, $x'_{k'} = k'/n$. Notice that the right-hand side of (19) does not depend on ℓ or ℓ'. Summing (19) over ℓ,ℓ' where $1 \le \ell,\ \ell' \le tn^{1/2} = n^{1/2-\alpha}$ leads to

$$\sum E\,\{Y_n(k,\ell)[Y_n(k',\ell')-\delta]\} = O\left[g(x_{k-1},x'_{k'-1})n^{-2-2\alpha}\right] \quad .$$

Finally, replacing the sum over k,k' where $sn \le k',k,\ k' + k \le n$ by the double integral of $g(x,x')$ over the region $s \le x,x',\ x + x' \le 1$, produces the desired conclusion.

References

1. M. E. Hellman, "A Cryptanalytic Time-Memory Trade-Off," IEEE Transactions on Information Theory, Vol. IT-26, No. 4, July 1980.

2. M. E. Hellman and J. M. Reyneri, "Drainage and the DES," Advances in Cryptology: Proceedings of Crypto '82, Plenum Press (New York, 1983).

3. J. J. Quisquater, "Some DES Cycling Results," Crypto '86.

4. B. S. Kaliski, R. L. Rivest, and A. T. Sherman, "Is DES a Pure Cipher," Advances in Cryptology: Proceedings of Crypto '85, Springer-Verlag (Berlin Heidelberg, 1986).

5. B. Harris, "Probability Distributions Related to Random Mappings," Annals of Mat. Statistics, 31 (1959), 1045-1062.

6. P. W. Purdom and J. H. Williams, "Cycle Length in a Random Function," Transactions of the American Mathematics Society, 133 (1968), 547-551.

7. P. G. Pittel, " On Distributions Related to Transitive Closures of Random Finite Mappings," Annals of Probability, Vol. II, No. 2 (1983), 428-441.

8. Yu. L. Povlov, "A Case of the Limit Distribution of the Maximum Size of a Tree in a Random Forest," Mat. Zametki, Vol. 25, No. 5 (1979), 751-760.

9. I. B. Kalugin, "Branching Processes and Random Mappings of Finite Sets," Mat. Zametki, Vol. 34, No. 5 (1983), 757-771.

10. I. B. Kalugin, "Characterization of Random Mappings," Mat. Zametki, Vol. 39, No. 3 (1986), 424-430.

11. J. W. Moon, Counting Labelled Trees: A Survey of Methods and Results, Canadian Mathematical Monographs, No. 1, 1970.

12. W. Feller, An Introduction to Probability Theory and Its Applications, Vol. I, John Wiley, New York (1968).

Fast Spectral Tests for Measuring Nonrandomness and the DES

Frank A. Feldman
Department of Physics, Suffolk University
Boston, MA 02114

Abstract — Two spectral tests for detecting nonrandomness were proposed in 1977. One test, developed by J. Gait [1], considered properties of power spectra obtained from the discrete Fourier transform of finite binary strings. Gait tested the DES [10,11] in output-feedback mode, as a pseudorandom generator. Unfortunately, Gait's test was not properly developed [3,4], nor was his design for testing the DES adequate.

Another test, developed by C. Yuen [2], considered analogous properties for the Walsh transform. In estimating the variance of spectral bands, Yuen assumed the spectral components to be independent. Except for the special case of Gaussian random numbers, this assumption introduces a significant error into his estimate.

We recently [3,4] constructed a new test for detecting nonrandomness in finite binary strings, which extends and quantifies Gait's test. Our test is based on an evaluation of a statistic, which is a function of Fourier periodograms [5]. Binary strings produced using short-round versions of the DES in output-feedback mode were tested. By varying the number of DES rounds from 1 to 16, it was thought possible to gradually vary the degree of randomness of the resulting strings. However, we found that each of the short-round versions, consisting of 1, 2, 3, 5 and 7 rounds, generated ensembles for which at least 10% of the test strings were rejected as random, at a confidence level approaching certainty.

A new test, based on an evaluation of the Walsh spectrum, is presented here. This test extends the earlier test of C. Yuen. Testing of the DES, including short-round versions, has produced results consistent with those previously obtained in [3].

We prove that our measure of the Walsh spectrum is equivalent to a measure of the skirts of the logical autocorrelation function. It is clear that an analogous relationship exists between Fourier periodograms and the circular autocorrelation function.

1. Introduction

Kolmogorov [7] and Chaitan [8] have established a theory of the information content of strings, which has been used to define random strings. Particular tests to detect certain irregularities of pseudorandom strings are presented in Knuth [9]. Fast spectral tests, in this spirit, have since been proposed. These tests evaluate either the fast Fourier transform (FFT) [6], or the fast Walsh transform (FWT) [12], of a finite test string. Both of these kinds of spectral tests were first proposed in 1977. One, presented by

J. Gait [1], examined the Fourier power spectra. The other, presented by C. Yuen [2], examined analogous properties for the Walsh transform. In the form presented, neither of these early tests can be compared directly with the output of the tests described by Knuth.

The purpose of Gait's paper was to test the DES, in output-feedback mode, as a pseudorandom generator. A significant part of his paper was devoted to the development of a power spectrum test; this test was applied to binary strings of length 2^{15} bits generated by the DES. Each test string can be specified by approximately 2^{14} spectral components. Nevertheless, on the basis of twenty-seven sample points taken from the power spectrum of one such sequence, the DES output was judged to be random. Actually, one can show that the graphic display presented by Gait is too flat, and thereby offers strong evidence for rejecting this test string as random [3,4].

The test proposed by C. Yuen considered analogous properties for the discrete Walsh transform. Yuen tested output from two kinds of pseudorandom generators: a Gaussian generator, and a generator of uniformly distributed "reals" ranging from 0 to 1. Yuen recognized — as Gait did not — that "... a spectrum estimate that looks too flat is as suspect as one not flat enough." Yuen also recognized that estimates for individual contributions to the Walsh power spectrum are not consistent. To circumvent this difficulty he chose to consider bands of spectral contributions. In his estimate of the variance for a band, Yuen assumed the spectral contributions to be statistically independent. This assumption is valid for the case of random numbers with a Gaussian distribution. However, in the general case, this assumption violates the *Parseval constraint* [5] on the spectrum and introduces a significant error into ones estimate of the variance.

We recently [3,4] proposed a new test for detecting nonrandomness in finite binary strings. Our test extends and quantifies Gait's test. We tested binary strings of length 2^{15} bits which were produced using short-round versions of the DES, in output-feedback mode. By varying the number of DES rounds from 1 to 16, it was thought possible to gradually vary the degree of randomness of the resulting strings. We found that for ensembles of test strings generated by short-round versions, consisting of 1, 2, 3, 5 and 7 rounds, each ensemble yielded test strings for which at least 10% were rejected as nonrandom at a confidence level approaching certainty.

We now propose a similar test based on an evaluation of the Walsh spectrum. This test is an extension of Yuen's test. We note that the fast Walsh transform (FWT) can be obtained from the fast Fourier transform (FFT) by setting all sines to 0 and all cosines to 1. It is usual to find a program block in the FFT which reorders the indices — by combining first a bit reversal, followed by Gray coding of the indices [12]. If one deletes this block, one obtains a particular representation of the Walsh transform which has the various designations: natural ordered Walsh transform, Walsh-Hadamard transform, and Hadamard transform. The FWT is at least four times faster than the FFT, and demands half the memory. We use the symmetry properties of the Hadamard transform to derive recursion relations which facilitate the computation of certain expectations: the expectations for estimating powers and products of Walsh periodograms. Thereby, the estimates of the mean and the variance called for in our test can be computed both easily and with precision.

2. Test for Nonrandomness

Our test computes a statistic, which is dependent on a finite length binary string. This statistic is compared with the value expected for a random string. The input string is interpreted as nonrandom if the computed value of the statistic differs too much from the expected value.

TEST (For the r^{th} moment; where $r = 4$, or $r = 6$)

INPUT: A string $x = x_0, \ldots, x_{n-1}$ of length $n = 2^k$.

PARAMETERS:

INTEGERS: $r = 4$, or $r = 6$. And $n = 2^k$, where n is sufficiently large [3,4].

REAL: t, the desired significance level, with $0 < t < 1$.

OUTPUT: "May be Random" or "Not Random."

STEP 1:

Representing the bits of x, with the values 1 and -1, the fast Walsh transform (FWT), $\hat{x}$, of x is computed. The algorithmic running time of the FWT is $O(nlogn)$. The space requirement is $O(n)$.

STEP 2:

The computation of the r^{th} power for each of the n Walsh transform components

$$\hat{x}^r = (\hat{x})^r,$$

where the input parameter r specifies that the r^{th} *moment* is being tested, and r is an *even* integer greater than 2.

STEP 3:

The computation of the statistic D_r, where

$$D_r = \sum_{k=0}^{n-1} (\hat{x}_k^r - m_r)/v_r. \tag{2.1}$$

The values assigned to m_r and v_r are defined in Section 4, by eq. (4.2) and (4.3).

STEP 4.

The decision is made "Not Random" or "May be Random."

The null hypothesis H_0 — that the input $x_0 \ldots x_{n-1}$ is random — is rejected at a level of significance, determined by the input t, if the integral

$$\frac{1}{\sqrt{2\pi}} \int_{-D_r}^{D_r} \exp\left(-\frac{y^2}{2}\right) dy \tag{2.2}$$

is less than $1 - t$.

3. Properties of the Walsh Transform

The Walsh transform is an orthogonal transformation of $n = 2^k$ variables. In *natural order* this transformation is effected through the use of a *Hadamard matrix.* For order 2, the symmetric Hadamard matrix is defined as

$$H[2] = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Higher order Hadamard matrices can be defined recursively as the direct product of lower order Hadamard matrices. Thus, for example

$$H[2n] = \begin{pmatrix} H[n] & H[n] \\ H[n] & -H[n] \end{pmatrix}$$

We prefer to define the Hadamard matrix, for order $n = 2^k$, in terms of its matrix elements. First, let the matrix indices s and t be represented in binary form as $(s_{k-1} \ldots s_0)_2$ and $(t_{k-1} \ldots t_0)_2$. Then

$$H_{s,t} = \prod_{p=0}^{k-1} (-1)^{s_p t_p}. \tag{2.1}$$

The Walsh transform of $x_0, \ldots, x_{n-1}$ is represented as

$$\hat{x}_s = \sum_{t=0}^{n-1} H_{s,t} x_t.$$

We show below that the *inverse* transformation is

$$x_s = \frac{1}{n} \sum_{t=0}^{n-1} H_{s,t} \hat{x}_t.$$

Five basic properties which follow from the above definitions are:

PROPERTY 1 (symmetry)

$$H_{s,t} = H_{t,s}$$

PROPERTY 2 (summation)

$$\sum_{s=0}^{n-1} H_{s,t} = n\delta(t),$$

where $\delta(s)$ is the *unit impulse* [5] function.

PROPERTY 3 (product)

$$H_{s,t}H_{s,p} = H_{s,t\oplus p},$$

where $t \oplus p \overset{\text{def}}{=} (t_{k-1} \oplus p_{k-1} \dots t_0 \oplus p_0)_2$, and $\oplus$ is the "xor" operation (addition modulo 2). For example $t \oplus t = 0$.

Proof:

$$\begin{aligned} H_{s,t}H_{s,p} &= \prod_{q=0}^{k-1} (-1)^{s_q(t_q \oplus p_q)} \\ &= H_{s,t\oplus p}. \end{aligned}$$

PROPERTY 4 (orthogonality)

$$\begin{aligned} \frac{1}{n}\sum_{t=0}^{n-1} H_{s,t}H_{t,p} &= \delta(s \oplus p) \\ &= \delta(s-p). \end{aligned}$$

PROPERTY 5 (*logical* shift)

$$H_{s,t}\hat{x}_t = \sum_{p=0}^{n-1} H_{t,p}x_{p\oplus s}.$$

Proof: $H_{s,t}\hat{x}_t = \sum_{p=0}^{n-1} H_{s,t}H_{t,p}x_p = \sum_{p=0}^{n-1} H_{t,s\oplus p}x_p$, where PROPERTY 3 has been used. And then, since $s \oplus p$ is simply a permutation of the integers $(0, \dots, n-1)$ over the range of the "dummy index" p, a relabelling of this index yields the sum $\sum_{p=0}^{n-1} H_{t,p}x_{p\oplus s}$.

We define Walsh periodograms as

$$I_s = (1/n)\hat{x}_s^2.$$

A sequence known as the *logical* autocorrelation function is defined as

$$\tilde{a}_s = \frac{1}{n}\sum_{t=0}^{n-1} x_t x_{t\oplus s}.$$

Theorem 1.
$\tilde{a}_0, \dots, \tilde{a}_{n-1}$ and $I_0, \dots, I_{n-1}$ are Walsh transform pairs [12]. That is

$$I_s = \sum_{t=0}^{n-1} H_{s,t}\tilde{a}_t,$$

and

$$n\tilde{a}_s = \sum_{t}^{n-1} H_{s,t}I_t.$$

Corollary (Parseval's theorem for the Walsh transform)

$$\sum_{s=0}^{n-1} I_s = n\tilde{a}_0\,, \text{ or equivalently } \sum_{s=0}^{n-1} \hat{x}_s^2 = n\sum_{s=0}^{n-1} x_s^2.$$

A degenerate case of Parseval's theorem occurs with the restriction $x_s \in [-1, 1]$, in that it yields a *deterministic* value for the sum of the squared Walsh coefficients. And for this case

$$\sum_{s=0}^{n-1} \hat{x}_s^2 = n^2.$$

Consider the following important example. When testing n random bits, it is usual to map bit values from $[0, 1]$ onto the values $[-1, 1]$. And for this case, Parseval's theorem tells us that the summed value for the "power spectrum" is *identical* for all test strings. This result is referred to as *deterministic* for the following reasons: First, the summed value is n^2 for any test sequence, independent of its distribution. Second, assume the set $x_0, \ldots, x_{n-1}$ to be a set of random variables, whose values are either 1 or -1. Let this set be assigned an arbitrary joint probability distribution. Yet, not only is the ensemble average for the summed "power spectrum" equal to n^2, but in addition the variance of this sum is zero. Thus, the induced probability distribution for the summed "power spectrum" degenerates into a singular distribution known as a *Dirac delta function*.

Such is also the case for the FFT. One can choose the option of testing *spectral bands*, as in [2]. However, to be in the asymptotic region for which the central limit theorem holds [3], one may have to increase the length n of the test strings. An alternative approach will be developed in the next section. However, the groundwork will be developed below.

Our main contribution, in this section, is presented in the form of two theorems. These two theorems will be stated below in the context of the FWT and the logical autocorrelation function $\tilde{a}$. However, both of these theorems are also valid in the context of the FFT, with the replacement of the FWT by the FFT, the Walsh periodogram by the Fourier periodogram [5], and the logical autocorrelation function by the circular [5] (positively wrapped [6]) autocorrelation function. This last replacement is effected by changing $s \oplus t$ to $(s + t) \bmod n$. Both of these theorems can be proved by means of a straightforward application of the five basic transform PROPERTIES listed earlier in this section.

Theorem 2.

$$\sum_{t=0}^{n-1} H_{s,t} I_t^2 = n \sum_{t=0}^{n-1} \tilde{a}_t \tilde{a}_{t\oplus s}. \tag{3.1}$$

Corollary

$$\sum_{s=0}^{n-1} I_s^2 = n \sum_{s=0}^{n-1} \tilde{a}_s^2. \tag{3.2}$$

Theorem 3.

$$\sum_{t=0}^{n-1} H_{s,t} I_t^3 = n \sum_{t=0}^{n-1}\sum_{p=0}^{n-1} \tilde{a}_t \tilde{a}_{t\oplus s\oplus p} \tilde{a}_p. \tag{3.3}$$

Corollary

$$\sum_{s=0}^{n-1} I_s^3 = n \sum_{s=0}^{n-1}\sum_{t=0}^{n-1} \tilde{a}_s \tilde{a}_{s\oplus t} \tilde{a}_t. \tag{3.4}$$

Discussion of these two theorems is postponed to the next section.

4. Ensemble Averages

In testing for randomness an *a priori* requirement is that the type of randomness be defined. This requirement is met by considering the string variables to be random variables and then assigning a joint probability distribution for these variables. The theorems presented in the previous section can be applied to these random variables. It is possible to interpret these theorems in terms of ensemble averages. Consider, for example, the corollary of theorem 2. We shall restrict our attention to *binary* test strings of length 2^k — where for convenience, bits will be represented as -1 and 1. Taking expectations, yields the relation

$$E[\sum_{s=0}^{n-1} \hat{x}_s^4] = n^3 + n^3 \sum_{s=0}^{n-1} E[\tilde{a}_s^2], \tag{4.1}$$

where use has been made of the relation $\tilde{a}_0 = 1$. Thus, the expected sum of the fourth power of the spectral components is related to the expected sum of the square of the terms making up the skirts of the logical autocorrelation function. It is clear that, for the FFT, a similar relation exists between the squared periodograms and the circular autocorrelation function.

We now restrict our attention to *symmetric Bernoulli sequences.* That is, sequences $x_0, \ldots, x_{n-1}$ which are independent and identically distributed with $Pr(x_k = 1) = \Pr(x_k = -1) = 1/2$, for $k = 0, \ldots, n-1$. Our goal is to numerically evaluate terms of the form

$$m_r = E[\sum_{s=0}^{n-1} \hat{x}_s^r] \tag{4.2}$$

and

$$v_r = E[(\sum_{s=0}^{n-1} \hat{x}_s^r)^2] - m_r^2 \tag{4.3}$$

for $r = 4$ and $r = 6$. These two expectations are called for as parameters in STEP 3 of our test for nonrandomness.

Combining ,the symmetry of the distribution with the structure of the Walsh transform yields the equivalences

$$m_r = nE[\hat{x}_0^r], \tag{4.4}$$

and

$$v_r = nE[\hat{x}_0^{2r}] + n(n-1)E[\hat{x}_0^r\hat{x}_1^r] - n^2(E[\hat{x}_0^r])^2. \tag{4.5}$$

Thus to complete the numerical evaluation of m_r and v_r it is necessary to compute expectations of the form $E[\hat{x}_0 s]$ and $E[\hat{x}_0^s\hat{x}_1^s]$, where s is an even integer. This can be accomplished by means of the two recursion relations which are presented below.

First note that $E[\hat{x}_k^s] = E[\hat{x}_0^s]$ for $k = 1 \ldots n-1$ and $E[\hat{x}_0^{2s+1}] = 0$. We now use the notation $\hat{x}_k(n)$ to indicate the k^{th} transform of a sequence of n bits and note that for the trivial case of $n = 1$

$$E[\hat{x}_0^s(1)] = 1. \tag{4.6}$$

One can then show that

Recursion Relation 1.

$$E\left[(\hat{x}_0(2n))^{2r}\right] = \sum_{s=0}^{r} \binom{2r}{2s} E\left[(\hat{x}_0(n))^{2(r-s)}\right] E\left[(\hat{x}_0(n))^{2s}\right] \tag{4.7}$$

Recursion Relation 2.

$$E\left[(\hat{x}_0(2n)\hat{x}_1(2n))^{2r}\right] = \sum_{s=0}^{2r} \binom{2r}{s} (-1)^s E\left[(\hat{x}_0(n))^{4r-2s}\right] E\left[(\hat{x}_0(n))^{2s}\right] \tag{4.8}$$

These two relations can be proved by representing $\hat{x}_0(2n) = \hat{y}_0(n) + \hat{z}_0(n)$ where $\hat{y}_0(n) = \sum_{s=0}^{n-1} x_s$ and $\hat{z}_0(n) = \sum_{s=n}^{2n-1} x_s$, and then using the binomial expansion prior to taking their expections. To derive recursion relation 2, a similar device is used to re-express $\hat{x}_1(2n)$ as the sum of two statistically independent terms.

We list the explicit solution to the first recursion relation for two terms:

$$E[\hat{x}_0(n)^2] = n, \quad \text{and} \quad E[\hat{x}_0(n)^4] = n + 3n(n-1).$$

5. Statistical Results

In this section we report the results obtained from a series of statistical tests which we performed on binary test strings of length 2^{13} bits. Both the DES, and short-round versions of the DES, run in output-feedback mode, were used to generate our test strings.

By thus varying the number of rounds from 1 to 16, it was thought possible to gradually vary the degree of randomness of the resulting strings. For each of these 16 gradations we generated an *ensemble* of 10 test strings.

Plaintext consisting of all zeroes was used to initiate generation of all the test strings. The same 10 keys were used for each ensemble. These keys were generated by using the DES, in output-feedbak mode, with a seed of all zeroes, and the key $FFFFFF00FF000000_{16}$. The k^{th} output block of ciphertext was subsequently used as the key for generating the k^{th} test string of each ensemble.

Three tests measuring nonrandomness were performed on the test strings. If any of these tests indicated the nonrandomness of a test string at the 5% level, then the test string was flagged and the characterizing parameters for all three tests were printed.

The first test measured uniformity of distribution with respect to bit values. Each test string was partitioned into 2^{11} ordered sets — each set consisting of four bits. To measure uniformity of distribution, the chi-square statistic was computed with respect to the 16 possible realizations of four bits. When the chi-square value is greater than 24.996 (for 15 degrees of freedom) the string distribution is significantly nonrandom at the 5% level.

The second, and third tests were our tests for evaluating the second, and third moments ($r = 4$ and $r = 6$). If the characterizing parameter, for either of these tests, is greater than 1.960, then the input string is significantly nonrandom with respect to that test, at the 5% level.

Our results are presented in Table 1 below.

For the one round truncation of the DES, a seed of all zeroes was used to generate the strings. As a consequence, its output is limited to a repetition of two 64 bit blocks. The first block necessarily contains zeroes in its 32 odd positions, and a mixture of zeroes and ones in its even positions. The second block, also by necessity, contains 64 zeroes. All the pairs of blocks which follow, are a repetition of the first two blocks. Thus, one expects the characterizing parameters for each nonrandom test to reach near maximum values. Table 1 satisfies this expectation for all of the test strings in this ensemble.

Every output block generated by the two-round version of the DES is uniquely encrypted. Yet, all but three of the ten test strings generated, were flagged at the 5% level of significance, with one string rejected as random by both of the spectral tests, but not by the chi-square test. However, the majority of test strings was found to be nonrandom at a level approaching certainty.

For the three-round version of the DES, all ten strings were flagged at the 5% level of significance by both of the spectral tests. For nine of the test strings, the random hypothesis was rejected at a level approaching certainty. Six of the strings were rejected at the 5% level by the chi-square (two of these were at a level approaching certainty).

Using four or more rounds, the number of strings flagged as nonrandom, at the 5% level, were within a range acceptable for random strings. However, the five round version generated a single test string which was found to be nonrandom by all three tests, at a

Table 1

String label	χ^2	$r = 4$	$r = 6$
1 Round			
1	7872.000	0.289E+05	0.560E+07
2	6720.000	0.196E+05	0.245E+07
3	6720.000	0.237E+05	0.380E+07
4	6720.000	0.199E+05	0.246E+07
5	7744.000	0.203E+05	0.261E+07
6	6784.000	0.202E+05	0.260E+07
7	6336.000	0.208E+05	0.278E+07
8	8256.000	0.251E+05	0.430E+07
9	6720.000	0.206E+05	0.278E+07
10	6976.000	0.199E+05	0.249E+07
2 Rounds			
2	143.906	144.	640.
4	688.000	1080.	7150.
6	116.984	139.	553.
7	45.078	67.3	243.
8	185.406	202.	894.
9	99.453	55.2	151.
10	12.891	12.7	21.7
3 Rounds			
1	30.406	18.3	33.9
2	28.047	23.7	66.3
3	19.609	12.8	24.8
4	25.047	32.7	77.7
5	25.187	46.3	116.
6	21.297	14.1	23.7
7	21.281	13.1	23.7
8	15.672	3.49	2.94
9	100.172	125.	527.
10	49.687	22.6	44.1
4 Rounds			
8	25.844	0.472	0.731
5 Rounds			
1	12.687	3.31	2.56
3	34.094	0.426	0.586
9	44.109	9.21	17.6
6 Rounds			
1	10.594	2.43	1.72
7 Rounds			
5	33.656	12.2	23.6
7	9.812	2.33	1.56
8 Rounds			
4	25.812	1.16	1.22
7	13.437	1.45	2.02

Table 1 (cont.)

String label	χ^2	$r = 4$	$r = 6$
9 Rounds			{ None flagged at 5% level}
10 Rounds			
4	10.250	−2.18	−2.13
Rounds 11,12,13			{None flagged at 5% level}
14 Rounds			
4	13.922	−2.17	−2.08
15 Rounds			
2	19.953	−2.46	−1.62
4	27.641	0.2537	−0.114
7	7.047	1.76	2.20
10	26.875	0.999	0.823
DES (16 Rounds)			{None flagged at 5% level}

level approaching certainty. One test string generated by the seven-round version was found to be nonrandom by both the spectral tests at a level approaching certainty, while the chi-square test accepted this string as random at less than the 0.5% level. Thus, probabilistic considerations force one to conclude that the ensembles generated by the five and the seven round versions of the DES are not random. Ensembles generated by eight or more rounds, were found to have good statistical properties.

Our spectral test for $r = 4$ appears to be a good complement to the chi-square test. Looking at test strings generated by four or more rounds of the DES, one observes little overlap between the chi-square test and the spectral tests. Of the fifteen strings flagged at the 5% level, five were flagged by the chi-square test alone; and six which were not flagged by this test, were flagged by the $r = 4$, spectral test. The overlap between the two spectral tests — though not total — was high.

We note that when we previously used our spectral test based on the FFT [3] to test the DES round by round we used the same set of keys to generate our test ensembles. Our test strings, however, were of length 2^{15} bits. (This is the length originally chosen by Gait in his test [1].) When we compare the output of these tests with our new results, they are essentially the same string for string except that for the longer test strings, the significance level of the rejected strings is generally much higher. This indicates that tests based on the FWT, and the FFT, give similar results; and also that the "bad" strings are key dependent, since they remain nonrandom when the length of the test string is increased by a factor of four.

References

[1] J. Gait, "A New Nonlinear Pseudorandom Number Generator," IEEE Trans. on Software Eng., Vol. SE–3(5) pp. 359–363 (Sept. 1977)
[2] C. Yuen, "Testing Random Number Generators by Walsh Transform," IEEE Trans. on Computers, Vol. C–26(4) pp. 329–333 (April 1977)
[3] F. A. Feldman, "A New Spectral Test for Nonrandomness and the DES," submitted to IEEE Trans. on Software Engineering (July 1986)
[4] F. A. Feldman, "A New Spectral Measure of Nonrandomness," Suffolk University Technichal Report No. 5, (1987)
[5] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Prentice-Hall, Inc., Englewood Cliffs. New Jersey, 1975.
[6] A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishiing Company, Reading, Mass. 1974.
[7] A. Kolmogorov, "Three Approaches to the Quantitative Definition of Information," PROB. INFO. TRANSMISSION, Vol. 1, No. 1, Jan. 1965, 1–7.
[8] G. Chaitan, "On the Length of Programs for Computing Finite Binary Sequences," JACM (13), 1966, 547–569.
[9] D. Knuth, *The Art of Computer Programming; Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Reading, Mass. 1969.
[10] "Data Encryption Standard," FIPS PUB 46, National Bureau of Standards, Washington, D.C., Jan. 1977.
[11] H. Katzan, *The Standard Data Encryption Algorithm*, Petrocelli Books, Inc., New York, 1977.
[12] K. G. Beauchamp, *Applications of Walsh and Related Functions*, Academic Press, 1984.

OTHER CYCLING TESTS FOR DES

Jean-Jacques Quisquater & Jean-Paul Delescaille

Philips Research Laboratory Brussels
Avenue Van Becelaere, 2
B-1170 Brussels, Belgium

Abstract

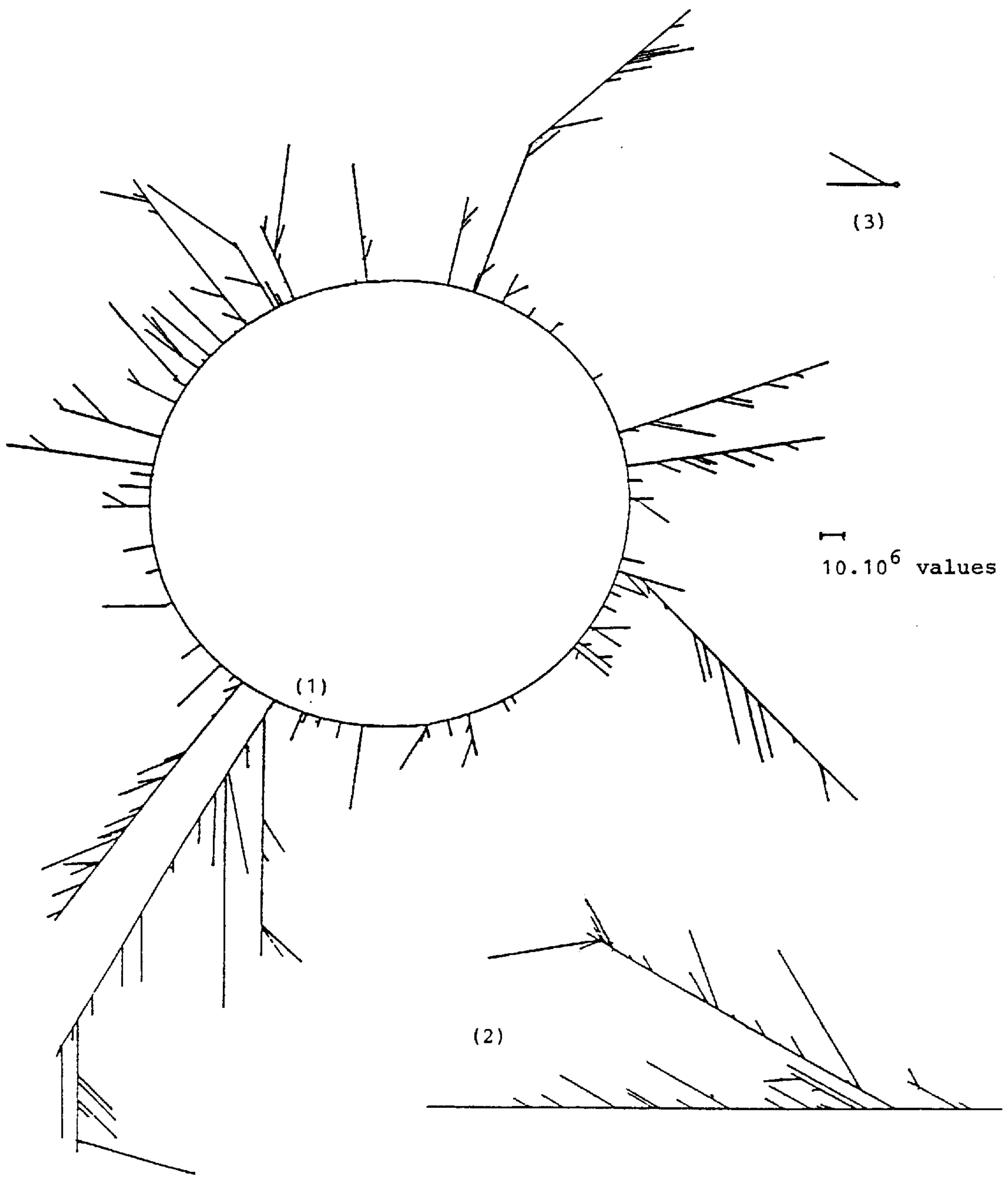
A very preliminary presentation of this paper was done at the rump session of CRYPTO '86 and kindly played by Gus Simmons (Sandia). The full paper will be complete description of some cycling experiments applied to DES using only very fast software for the computations. Using only software permitted many measures and tests.

The used version of DES has a speed of about 2 Mbits/sec for a given key and a speed of about 600 kbits/sec for a change of key and an encryption (this version is not fully optimized due to some problems with the cache of the computer). For these experiments we used the idle time of an IBM 3090 and a total of 2^{34} encryptions were performed. Many techniques were used in order to avoid duplicate computations and big files.

The cycling test we done is the following one: Given a "random" key and a fixed message, what is the cycle structure if the output is next used as key after some fixed transformation? For a given configuration, 3 cycles were found, the smallest having only about 60.000 elements. Many statistical tests were done (distribution of "distances" between typical points, cumulative distributions, lengths of tails before collision, etc). A total of 1900 initial points were tested. Each initial point belongs to one of the three found cycles: A very fast technique was used to identify the related cycle. The three cycles were drawn giving more visual informations about random mappings (see next page).

New techniques to find large cycles and collisions of tails were elaborored using the notion of *distinguished point* (an output of DES with some quickly tested property), a related concept for the use of an hashing function and a clever time-memory trade-off.

From a theoretical point of view, the relation with the theory of random mappings and the use of DES to obtain random numbers (see J. Gait) was done and a better view of "drainage" (see Reyneri and Hellman) was obtained. The use of distinguished points is also possible for the study of exhaustive machines (Hellman)


(3)
10.10⁶ values
(1)
(2)

A Crypto-Engine

George I. Davida
Frank B. Dancs

University of Wisconsin-Milwaukee
Milwaukee, WI 53201

Abstract

In this paper we present a design for a crypto-engine. We shall discuss the design and show the instruction set of this coprocessor and then show how this could be used to implement most of the known encryption algorithms. We will discuss why a coprocessor approach may be a better solution than adoption of specific encryption algorithms which can be broken or decertified.

I. Introduction

With the ever increasing use of encryption techniques to safeguard data, a need has become apparent for special hardware to implement these schemes due to the inefficiency of software methods. Special hardware such as DES chips have the draw back of being useful for only one encryption technique. Many have discussed the weekness of the method and site potential problems with the sboxs. Furthermore, if DES is not certified as the standard for encryption, many existing chips will become obsolete.

Taking this into consideration, a hardware design that could increase the efficiency of computation while allowing versatility for many existing encryption algorithms and possibly future ones has been considered. Much like a floating point coprocessor, we have designed an encryption coprocessor. With basic instructions common to the variety of algorithms used today, the design presented in this paper facilitates implementing of most encryption algorithms.

The design of this coprocessor has been based on the Motorola 68000[1] coprocessor protocol. It has been implemented in software for testing purposes. Implementation details will also be discussed later in the paper.

The research reported in this paper was supported in part by NSF grant DCR-8504620.

II. Need For a Coprocessor

With chip prices coming down, replacing software tools with hardware tools has become more practical. The problem with hardware is that it may not always be as flexible as one would want. Take for example the DES chips that are on the market. Although they are excellent for preforming their assigned tasks, they can only preform one task and that is encrypting data with the DES standard. If this standard is broken or decertified as the standard, these chips would be obsolete. Replacement costs for this specialized hardware could be expensive.

Unlike other types of hardware, encryption hardware has the drawback of potentially becoming useless with the ever possible chance that the algorithm the hardware implements is broken. Other types of hardware could become obsolete with the advent of a new and better device, however; the old hardware is still useful.

The other disadvantage of a standard encryption chip is that standards can be changed. A specialized chip can only implement one encryption technique. If the standard would change then those using the standard would not be able to communicate with those using the old standard. Since reprogramming is not possible, this desired versatility cannot be achieved by these specialized chips. After all, what is the difference between a calculator and a computer? It is not the greater capability of the computer or its faster processing. It is the versatility of the computer brought about by the ability to be programmed for many different applications.

With this in mind a coprocessor with special instructions that would be useful for encryption appears to be a better solution. If a standard encryption technique is decertified a coprocessor could easily be reprogrammed. A new technique could easily be installed. Not only could a new standard be adopted without any hardware changes, different techniques could be used simultaneously for different applications. Furthermore, techniques could be altered to meet specific requirements of local environments. The advantage over just pure software implementations of the encryption techniques is, of course, special instructions implemented in hardware would bring about a speed advantage.

III. Coprocessor Design

To make a hardware device that can implement most of the known algorithms of today and still leave room for tomorrows, we need to consider what are the basic operations of encryption algorithms. There are two basic operations that are used for encryption. They are substitution and transposition [2]. Most crytographic techniques use a combination of both.

Substitution operations can be table lookup, many types of arithmetic operations such as multiplication, exponentiation, etc. Transposition operations on the other hand are of the form, permutation on the bits of a word, shift registers, and modular arithmetic, where the permutation is of the entire message space.

With this in mind, we propose a set of encryption primitives. The substitution instructions include all arithmetic in large register form, an actual table lookup and its supporting instructions, and three of the boolean operators, *and*, *or*, and *xor*. For the permutation operations, we included a bit permuter, three different shift operations, and the modular arithmetic operation.

IV. Design Details

The registers, illustrated below, that will be needed to support these instructions must be of a larger than normal size. For example, RSA uses numbers approximately of 200 decimal digits large. For this type of arithmetic, there are four large registers, 1024 bits each. This will allow for roughly 300 decimal digits. These registers will support all the arithmetic instructions. There will also be the need for smaller registers for substitution and permutation operations. There are five 128 bit registers. One of these registers, the general register, can be addressed as four 32 bit registers, two 64 bit registers, or the entire 128 bit register. Finally, there are 16 tables consisting of 256 bytes each supporting the table lookup operation.

Register Layout of the Encryption Coprocessor

The larger register layout

1024 bits L1 Large Purpose register
1024 bits L2 Large Purpose register
1024 bits L3 Large Purpose register
1024 bits L4 Large Purpose register (ML) Modules constant

General register, key register and modular constant layout (128 bits)

Ghl(high.left)	Ghr(high.right)	Gll(low.left)	Glr(low.right)
128 bits K0 key register			
128 bits K1 key register			
128 bits K2 key register			
128 bits MS Small Modular constant register			

Sbox Array

Array For Slice 1

byte 0	byte 1	byte 2	byte 3	. . .	byte 255

Array For Slice 2

byte 0	byte 1	byte 2	byte 3	. . .	byte 255

. . .

. . .

. . .

Array For Slice 16

byte 0	byte 1	byte 2	byte 3	. . .	byte 255

The instructions listed in Appendix A are the instructions that we have considered necessary to fulfill the requirements of our list of basic operations. The instructions listed in the Appendix follow these rules of use. Those that have <reg>, <greg>, or <lreg> can support the register direct addressing mode on any register, the general register only, and one of the large registers only, respectively. The instructions that have <reg1> or <reg2> as their operands have the following addressing modes: register direct, register indirect, memory immediate, and memory indirect. Furthermore, the indirect addressing mode using the main processors registers *a0*, *a6*, and *a7* has the following subdivisions: register indirect, register indirect with auto-increment, register indirect with auto-decrement, and register indirect with index.

V. Detailed Description Of Some Selected Instructions

The brief description of each instruction presented in the Appendix may not give enough information on some of the more complex instructions, and may not completely show the versatility of them.

The sbox substitution is probably the most complex operation. The *esbox* instruction can do substitution on a number of bit combinations. The programmer can choose from 8, 6, or 4 bits into the substitution and 8, 6, or 4 bits out of the substitution. The number of input bits does not have to be the same as the number of output bits. To initialize this we will need the *einitsbox* instruction. This will set up the *esbox* instruction for operation with one of these combinations of input bits and output bits. We will refer to a string of bits that will be used for one atomic substitution, either 8, 6, or 4 bits in size, as a substitution word.

As seen by the register layout diagram, there are 16 sbox arrays, each having 256 bytes of storage. Each of these arrays are to be used for one substitution. That is, one substitution word will be an index to one of these sbox arrays. When the substitution word indexs one of the bytes in the array, 8 bits, 6 bits, or 4 bits will be used depending on the number of bits that has been set up by the *einitsbox* instruction for output.

All of the memory in the sbox array will not be utilized for every substitution combination. For example, if a programmer where to initialize this operation to 4 bits in and 8 bits out with a 64 bit register, all of the 16 sbox arrays would be used, however, only 16 bytes out of each array would be used.

To load the sbox arrays the *eldsbox* instruction will be used. The number of the array will be needed as well as the number of bytes in that array. In the example above, there will be 16 *eldsbox* instructions needed to load all of the arrays. Each of these instruction will specify *n-1* as the number of bytes to load and *m* for the slice number. The only address mode that will be allowed is memory indirect. This will allow the programmer to initialize their sbox arrays somewhere in memory with a label, and use that label with the instruction.

There are two permutation instructions, *eperms* and *epermd*, representing control from the source and control for the destination. These will be used for two different types of permutations. The *eperms* instruction will allow permutation of one bit to multiple bits, while, the *epermd* instruction, will allow multiple bits to be permuted to one bit. Below is a diagram that should clear up the two different permutation instructions. Note, only four bits of the general register is used, and the first four bytes of the *l1* registers are used. In the *l1* registers we will store the following first four control bytes, these will control the first four bits in the general register signified by a letter in the alphabet.

4	1	2	4

This is what is stored in the first four bits of the original general register, *g*.

A	B	C	D

Permuting the original with the instruction: *eperms l1,g*

D	A	B	D

Permuting the original with the instruction: *epermd l1,g*

B	C	0	$A \oplus D$

The example above points out the situation for the *epermd* instruction where no bit was assigned to destination of the third bit location. In this case, the third location obtains the value zero. Furthermore, if two or more bits go to the same destination, the values of those two or more bits are *xor*ed together.

The *eperms* instruction would be used in something like the sbox expansion of DES, and the *epermd* instruction would be used in something like a linear shift register with a finite state machine.

VI. Details of Various Algorithm Implementation

The following implementations are written in the high level language of C and the program listings are found in Appendix B. The asm is a construct that allows a programmer to give an instruction directly to the assembler. After compiling this code the modified assembler can translate the coprocessor instructions into machine language. All of the coprocessors instructions are noted by italics. Note, some parts of main line, functions, and array initializations are omitted to conserve space.

DES[3] was an important consideration in the design of the coprocessor. As long as DES is the standard there will be a need to implement it.

A few points need to be noted for clarity. First, the sbox expansion is done like a standard DES sbox expansion. However, after the *xor* with the key, the bits must be permuted again to facilitate the b5b0b4b3b2b1 pattern. In other words, since the sbox substitution instruction of the coprocessor does a direct table lookup on the address that the six given bits generate, they must be permuted to follow the pattern that is necessary to follow the DES algorithm. This is because the algorithm calls for the first and last bit of the six bit substitution word be the index to the row and the middle bits be the index to the column. In the coprocessor the data is layed out continuously instead of having a row and column; thus, the last bit needs to be moved to the second bit. Since the large registers are capable of holding a 128 bit permutation, the sbox_expansion array will hold both permutations, which in turn is moved to the *l4* register.

The RSA implementation is pretty straight forward since the algorithm is nothing more than:

$$C = M^e \bmod n \quad \text{(encrypt)}$$
$$M = C^d \bmod n \quad \text{(decrypt)}$$

where *e* is the encryption key and *d* is the decryption key. This means that the only steps that need to be done are load one of the large registers with the data, do an exponential instruction with an automatic modular arithmetic with the *ml* register, and move the data from the large register back into memory.

The Pohlig-Hellman scheme is another exponential encryption technique; however, it could be implemented in two different ways. The basic algorithm is [4]

$$C = M^e \bmod p$$

$M = C^d \bmod p$

where *e* is the enciphering key and *d* is the deciphering key. The *p* could either be a prime or a GF(2^m) irreducible polynomial. Either way, it could be implemented quickly and easily.

The first implementation, where p is a prime, could be done exactly like the RSA listed in Appendix B, except ml would receive the prime value. To do the next implementation, do the exact implementation as the RSA example with these two changes. Put a irreducible polynomial in *ml*, and replace the following line:

```
    asm(" eexpml      l1,l2");
```

with

```
    asm(" eexpgfml    l1,l2");
```

VII. Remarks

Many other algorithms could be implemented. For example, the Shamir [5] Lagrange Interpolating Polynomial Scheme could be done with the arithmetic under Galois Fields instructions. All the necessary instructions are available to implement this scheme: addition, division and multiplication all in GF(2^m). Consider the irreducible polynomial $p(x) = x^3 + x + 1$; the corresponding binary representation would be 1011. This number would be put in the *ml* or *ms* register depending on the size of the integer arithmetic. This would then mean that all the arithmetic would be done under the Galois Field GF(2^3) with 1011 as the irreducible polynomial.

Another method for encryption could be implemented just as easily. The Block Ciphers with Subkeys [6] could be implemented with the basic arithmetic mod instructions. In this scheme the basic operations are inverse (*edivml*), multiplication (*emultml*), and addition (*eaddml*).

The software implementation was done on a Motorola 68010 based system. The 68000 protocol for coprocessor instruction identification is what is referred to as an F-line instruction. This F-line instruction will precede any coprocessor instruction. It will have the coprocessor identification number as well as other information. If the coprocessor doesn't exist in hardware, the processor will invoke a F-line emulation trap (vector 11). With this, we can trap the instruction in the kernel and start the emulation process. On a UNIX bsd 4.2 system this is just a quick addition in the trap.c code to capture the interrupt and a function call to the emulation routines. In the emulation routines, the first step is to check what the instruction is and call the appropriate emulation function.

The implementation of the Galois fields followed the Scott, Stafford and Peppard[7] algorithm for multiplication and the Davida and Litow[8] algorithm for inverse. Other implementations of arithmetic were done by methods that made the software easiest to write, not considering hardware problems.

Appendix A

Instruction Set

eadd <reg1>,<reg2>	The eadd instruction adds the source <reg1> to the destination <reg2> and stores the result in the

	destination <reg2>.
eaddms <regl>,<reg2>	
eaddml <regl>,<reg2>	The eaddms and eaddml instructions do the same addition mod the (ms) and (ml) registers respectively.
eaddgfms <regl>,<reg2>	
eaddgfml <regl>,<reg2>	The eaddgfms and eaddgfml instructions do the addition in the Galios Field $GF(2^m)$ using the (ms) and (ml) registers respectively to store the irreducible polynomial P(x).
esub <regl>,<reg2>	The esub instruction subtracts the source <regl> from the destination <reg2> and stores the result in the destination <reg2>.
esubms <regl>,<reg2>	
esubml <regl>,<reg2>	The esubms and esubml instructions do the same subtraction mod the (ms) and (ml) registers respectively.
esubgfms <regl>,<reg2>	
esubgfml <regl>,<reg2>	The esubgfms and esubgfml instructions do the subtraction in the Galios Field $GF(2^m)$ using the (ms) and (ml) registers respectively to store the irreducible polynomial P(x).
emult <regl>,<reg2>	The emult instruction multiplies the source <regl> to the destination(reg2) and stores the result in the destination <reg2>.
emultms <regl>,<reg2>	
emultml <regl>,<reg2>	The emultms and emultml instructions do the same multiplication mod the (ms) and (ml) registers respectively.
emultgfms <regl>,<reg2>	
emultgfml <regl>,<reg2>	The emultgfms and emultgfml instructions do the multiplication in the Galios Field $GF(2^m)$ using the (ms) and (ml) registers respectively to store the irreducible polynomial P(x).
ediv <regl>,<reg2>	The ediv instruction divides the source <regl> into the destination <reg2> and stores the result in the destination <reg2>.
edivms <regl>,<reg2>	
edivml <regl>,<reg2>	The edivms and edivml instructions do the same division mod the (ms) and (ml) registers respectively.
edivgfms <regl>,<reg2>	
edivgfml <regl>,<reg2>	The edivgfms and edivgfml instructions do the division in the Galios Field $GF(2^m)$ using the (ms) and (ml) registers respectively to store the irreducible polynomial P(x).
eexp <regl>,<reg2>	The eexp instruction takes destination <reg2> to the power of the source <regl> and stores the result in the destination <reg2>.
eexpms <regl>,<reg2>	

eexpml <reg1>,<reg2>	The eexpms and eexpml instructions do the same exponentiation mod the (ms) and (ml) registers respectively.
eexpgfms <reg1>,<reg2>	
eexpgfml <reg1>,<reg2>	The eexpgfms and eexpgfml instructions do the exponentiation in the Galios Field GF(2^m) using the (ms) and (ml) registers respectively to store the irreducible polynomial P(x).
eunsetsign	
esetsigned	These two instructions will set all arithmetic in an unsigned mode and set all arithmetic in a signed mode, respectively.
emod <reg1>,<reg2>	The emod instruction will do a modular arithmetic operation. The destination <reg2> mod source <reg1> and place the results in the destination <reg2>.
emov <reg1>,<reg2>	The emov instruction will move <reg1> to <reg2>. This can also be used to load registers with the different address modes available.
eldsbox #array,#bytes,_memory_location	The eldsbox instruction will load one of the 16 sbox arrays. It will load the amount of bytes in that table that is specified by the #bytes field. The only addressing mode allowed is memory indirect, thus the program will have to give the location of where the sbox array is stored in memory.
einitsbox #c1,#c2	The einitsbox instruction will initialize the control for the sbox instruction. The #c1 will be the number of bits inputed for each substitution in the sbox operation and #c2 will be the number of bits output for each substitution. These bits can have the values four, six and eight.
esbox <greg>	The esbox instruction does a sbox array lookup with the <greg> and stores the result in the same register.
eperms <lreg>,<reg>	The eperm instruction will do a permutation on a the destination <reg> with the large control register <lreg>. Each byte in the control register will control where each bit in the destination <reg> will come from.
epermd <lreg>,<reg>	The eperm instruction will do a permutation on a the destination <reg> with the large control register <lreg>. Each byte in the control register will control the destination of corresponding bit in the destination <reg>.
eand <reg1>,<reg2>	The eand instruction will do a bitwise *and* on the two registers given and stores the result in the destination <reg2>.
eor <reg1>,<reg2>	The eor instruction will do a bitwise *or* on the two registers given and stores the result in the destination <reg2>.

exor <reg1>,<reg2>	The exor instruction will do a bitwise *xor* on the two registers given and stores the result in the destination <reg2>.
elshiftl #num of bits,<reg>	
ershiftl #num of bits,<reg>	The elshiftl and ershiftl will do a left or right respectively logical shift on the register specified by the number of bits given.
elshiftc #num of bits,<reg>	
ershiftc #num of bits,<reg>	The elshiftc and ershiftc will do a left or right respectively circular shift on the register specified by the number of bits given.
elshifta #num of bits,<reg>	
ershifta #num of bits,<reg>	The elshifta and ershifta will do a left or right respectively arithmetic shift on the register specified by the number of bits given.

Appendix B

DES Source Listing

```
#include <stdio.h>

char    g[] = { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
                0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0};
unsigned int    k0[32];
unsigned int    k1[4];
char            initial[] = { /* initial permutation control values here */ };

char            final[] = { /* final permutation control values here */ };

char            sbox_expansion[] = {127, 126, 125, 124, 123, 122, 121, 120,
                        119, 118, 117, 116, 115, 114, 113, 112, 47, 42, 46,
                        45, 44, 43, 41, 36, 40, 39, 38, 37, 35, 30, 34, 33,
                        32, 31, 29, 24, 28, 27, 26, 25, 23, 18, 22, 21, 20,
                        19, 17, 12, 16, 15, 14, 13, 11, 6, 10, 9, 8, 7, 5,
                        0, 4, 3, 2, 1, 63, 62, 61, 60, 59, 58, 57, 56, 55,
                        54, 53, 52, 51, 50, 49, 48, 0, 31, 30, 29, 28, 27,
                        28, 27, 26, 25, 24, 23, 24, 23, 22, 21, 20, 19, 20,
                        19, 18, 17, 16, 15, 16, 15, 14, 13, 12, 11, 12, 11,
                        10, 9, 8, 7, 8, 7, 6, 5, 4, 3, 4,  3, 2, 1, 0, 31};

char            internal[] = { /* internal permutation control values here */ };

char pc_1[]  =        { /* key pc-1 permutation value here */ };

char pc_2[]  =        { /* key pc-2 permutation control values here */ };

char sbox[8][64] =     {{13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
                         1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
                         7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
                         2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11},

                        { /* sbox 7 */ }
                        { /* sbox 6 */ }
                        { /* sbox 5 */ }
                        { /* sbox 4 */ }
                        { /* sbox 3 */ }
                        { /* sbox 2 */ }
                        { /* sbox 1 */ }};

main(argc,argv)
int argc;
char *argv[];
{
FILE *fpin, *fpout, *fkey, *fopen();
int iterations;
int i,n,flag;

        /* open input and output files here */

        do_key(flag);

        asm(" eunsetsign   ");
        asm(" eldsbox _sbox,     #0,#63");
        asm(" eldsbox _sbox+64,  #1,#63");
        asm(" eldsbox _sbox+128, #2,#63");
        asm(" eldsbox _sbox+192, #3,#63");
        asm(" eldsbox _sbox+256, #4,#63");
        asm(" eldsbox _sbox+320, #5,#63");
        asm(" eldsbox _sbox+384, #6,#63");
        asm(" eldsbox _sbox+448, #7,#63");
        asm(" einitsbox #6,#4");
        asm(" emov    _initial,l1");
        asm(" emov    _final,l2");
        asm(" emov    _internal,l3");
        asm(" emov    _sbox_expansion,l4");

        while((n = getdata(fpin,flag,g)) > 0 ) {
                asm(" emov    _g,g");
                asm(" eperms  l1,g");
                for(iterations = 0; iterations < 16; iterations++) {
                        k1[0] = k0[t*2];
                        k1[1] = k0[t*2+1];

                        asm(" emov    g,k0");
                        asm(" emov    #0,gh");
                        asm(" emov    #0,gll");
                        asm(" eperms l4,gl");
                        asm(" exor    _k1,gl");
                        asm(" eperms l4,g");
                        asm(" emov    gh,gl");
                        asm(" esbox   gl");

                        asm(" eperms l3,gl");
                        asm(" emov    k0,gh");
                        asm(" exor    ghl,glr");

                        if(iterations == 15) {
                                asm(" emov    glr,gll");
                                asm(" emov    ghr,glr");
                        } else
                                asm(" emov    ghr,gll");
                }
                asm(" eperms l2,g");
                asm(" emov   g,_g");
                putdata(fpout,flag,8,g);
        }

}
/* getdata function */
/* putdata function */

do_key(flag)
        /* read in key and make 16 subkeys that are placed in the k0 array.
         * Note, this function will be done much like the main line */
```

RSA Source Listing

```
#include <stdio.h>

#define SIZE 128

unsigned int ml[32];
unsigned int l1[32];
char l2[SIZE];

main(argc,argv)
int argc;
char *argv[];
{
FILE *fpin, *fpout, *fkey, *fopen();
char c;
int n,size;

        /* open files here for data in, data out */

        read_key(stdin,ml);
        size = find_ml_size(ml);
        read_key(stdin,l1);

        asm(" eunsetsign   ");
        asm(" emov    _ml,ml");
        asm(" emov    _l1,l1");

        while((n = getdata(fpin,flag,size,l2)) > 0 ) {

                asm(" emov    _l2,l2");
                asm(" eexpml l2,l1");
                asm(" emov    l2,_l2");
                putdata(fpout,flag,n,l2);
        }

}

/* find_ml_size, determines the size of the modulus so we don't read more
 * into the large register than we have room for in the modulus */
find_ml_size(ml)

        /* body of routine */
```

References

1. Motorola Inc, *MC68020 32-bit Microprocessor User's Manual*, Prentice-Hall Inc., Englewood Cliffs (1985).
2. D. Denning, *Cryptography and Computer Security*, Addison Wesley, Reading, MA (1982).
3. NBS, "Data Encryption Standard," *National Bureau of Standards*, FIPS PUB 46, (Jan 1979).
4. S. C. Pohlig and M. E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance," *IEEE Transactions on Information Theory* IT-24(January 1978).
5. A. Shamir, "How to Share a Secret," *Communications of the ACM* 22 pp. 612-613 (November 1979).
6. G. I. Davida, L. D. Wells, and J. B. Kam, "A Database Encryption System with Subkeys," *ACM Trans. on Database Syst.* 6(2) pp. 312-328 (June 1981).
7. P. Scott, "A Fast VLSI Multiplier for GF(2^m)," *IEEE Journal on Selected Areas in Communications* SAC-4 pp. 62-65 (January 1986).
8. G. I. Davida and B. Litow, "Fast Parallel Inversion in Finite Fields," *CISS, The Johns Hopkins University*, (Davi85).

A Natural Taxonomy for Digital Information Authentication Schemes*

Gustavus J. Simmons
Sandia National Laboratories
Albuquerque, New Mexico 87185

There are two objectives that prompt the authentication of information; one is to verify that the information was, in all probability, actually originated by the purported originator, i.e., source identification, the other is to verify the integrity of the information, i.e., to establish that even if the message was originated by the authorized source, that it hasn't been subsequently altered, repeated, delayed, etc. These two objectives are normally treated in the theory of authentication as though they are inseparable, and will also be treated in that way here, although recent results by Chaum [1] demonstrating message integrity with source anonymity and by Fiat and Shamir [2], by Goldreich, Micali and Wigderson [3], and by others demonstrating verification of source identity with no additional information exchange show that the functions can in some instances be separated. The relevance of this comment to the subject matter of this paper is that it suggests that there may be a fourth independent coordinate in information authentication besides the three that will be discussed here. In spite of considerable effort, we have been unable to produce a convincing argument for or against this being the case, so we only mention the possibility for completeness.

In deference to the origins of the problem of authentication in a communications context, we shall refer to the authenticated information as the message and to the originator (of a message) as the transmitter. The message, devoid of any meaningful physical embodiment, is presented for authentication by a means that we call the authentication channel. In the simplest possible authentication scheme, the party receiving the message (the receiver) is also the one wishing to verify its authenticity; although, as we shall see, there are circumstances in which this is not the case. Authentication, however, is much broader than this communications based terminology would suggest. The information to be authenticated may indeed be a message in a communications channel, but it can equally well be data in a computer file or resident software in a computer; it can be quite literally a fingerprint in the application of the authentication channel to the verification of the identity of an individual [4,5] or figuratively a "fingerprint" in the verification of the identity

* This work performed at Sandia National Laboratories supported by the U. S. Dept. of Energy under contract no. DE-AC04-76DP00789.

of a physical object such as a document or a tamper sensing container [6]. In the broadest sense, authentication is concerned with establishing the integrity of information strictly on the basis of the internal structure of the information itself, irrespective of the source of the information.

In a series of papers [7,8,9,10], the present author has developed a mathematical model for the authentication channel in which sharp bounds are derived for the probability, P_d, that an i.e., either a message
sent by an unauthorized transmitter or else a message from an authorized transmitter that has been intercepted and either modified or replaced by a substitute message, to be accepted by the receiver as authentic. One of these bounds, first derived by Gilbert, MacWilliams and Sloan [11] in a slightly different setting than described above is;

$$P_d \geq \frac{1}{\sqrt{|\mathcal{E}|}} \tag{1}$$

where $|\mathcal{E}|$ is the total number of rules available to the transmitter/receiver for encoding states of the source into messages*. In 1985, we reported [10] a paradoxical situation in which the bound in (1) appeared to be violated. The explanation of this paradox is the source of the first dichotomy on which the taxonomy of authentication schemes is based.

Consider the following simple example: a transmitter, the Tx, wishes to communicate one of two equally likely instructions to a receiver, the Rx, say, to buy or sell a particular (agreed upon in advance) stock. The point being that precisely one bit of information is to be communicated from the Tx to the Rx, any other one-bit source, i.e., a toss of a fair coin, would serve equally well. The communication must take place over a publicly exposed communications channel on which a third party, known as the eavesdropper (or opponent), is listening. This channel is often described as a party line telephone when only secrecy is of concern, although a party line is not a completely adequate model for authentication as we will see later.

In the case of secrecy, i.e., the classical objective of encryption, it is assumed to be vital to the Tx's and Rx's interests that the eavesdropper(s) not know which instruction the Tx is sending to the Rx. The following simple protocol, known to cryptographers as Vernam or one-time key encryption, insures that the eavesdrop-

* Ideally we would call the information to be authenticated "messages" as is the practice in communications theory. However, if we adopt this convention we are forced to introduce terminology to designate the collection of sequences that can be sent through the channel. We would either have to coin a new word to designate the particular sequence of symbols sent to convey and authenticate a message -- none of which seem very natural -- or else use the cumbersome term "authenticated message". The term "authenticator", usually used in the sense of an authentication code word appended to a message, has too restricted a connotation for the general case. We have opted instead to use the term "message" to designate what is actually transmitted and to tolerate the rather artificial device that the information conveyed by a message is the state of a hypothetical source.

per's a priori and a posteriori probabilities of determining the Tx's instruction to the Rx are the same. In other words, eavesdropping on the communication channel doesn't help the opponent to deceive the receiver. In order to foil the eavesdroppers, the Tx and Rx agree in advance as to whether the Tx will speak truthfully when he says buy or sell, or whether he will lie in what he says. Since the information content of an instruction in this example is one bit, they must introduce at least one bit of equivocation (to the eavesdropper) about the Tx's actions. They do this by flipping a fair coin and using the outcome with the following protocol to decide on the Tx's course of action, i.e., whether he will lie or speak truthfully in his instructions to the Rx.

		Cipher		
		Buy	Sell	
Key	H	Buy	Sell	← Instruction
	T	Sell	Buy	

If heads comes up, the Tx will say "Buy" when he wants the Rx to buy and "Sell" when he wants the Rx to sell. If tails comes up, however, he will say "Sell" when he wants the Rx to buy, and so forth. It should be clear that by using the protocol in this way the eavesdropper will know no more about the actual (encrypted) instruction the Tx sent to the Rx as a result of listening in on their telephone conversation than he would have had he not listened at all. Such a cryptosystem was defined by Shannon to be <u>perfect</u> -- in the obvious meaning of the term that the a priori and a posteriori (after observing a legitimate cipher sent by the Tx) probabilities of the eavesdropper being able to determine the instruction are the same.

The cryptographic key in this simple example is the knowledge (shared by the Tx and Rx) of whether the Tx is telling the truth or not: encryption is the act by the Tx of either speaking truthfully or lying as determined by the key, the cipher is what the Tx says while decryption is the interpretation by the Rx of what the Tx actually meant, not necessarily what he said. It should be obvious that the Tx and Rx cannot reuse the key in this example to encrypt a second instruction since the eavesdropper could determine the key that was used (after the fact) by comparing the action taken by the Rx with the observed cipher (instruction ?). In order for this protocol to be secure for repeated communications, the Tx and Rx must secretly exchange in advance of any communication as much information in the form of keys (coin flips) as they later wish to communicate as encrypted instructions. This requirement for advanace secret key exchange and the associated key protection problem is a serious practical limitation to the usefulness of perfect encryption schemes, which, incidentally, are the only schemes whose cryptosecurity are presently mathematically demonstrable. The relevance of this example to the present discussion of authentication is that the security provided by this secrecy protocol is provable, i.e., it is independent of the computing power an opponent may bring to bear on breaking a cipher. Although it is not essential to the purposes of the present example, it should be pointed out that there are similar provably secure encryption protocols for arbitrary security requirements.

This example can be extended to illustrate the authentication of the one bit of information, i.e., of providing a means for the Rx to verify (with some calculable confidence) that a message actually came from the legitimate (authorized) Tx and not from someone impersonating the Tx and that it has not been altered subsequent to the legitimate Tx having sent it. In the secrecy protocol just described, if the opponent were in a position to not only listen to communications from the Tx but also to either send fraudulent ciphers (pretending to be the Tx) or else to intercept legitimate ciphers sent by the Tx and substitute others of his own devising then he could be certain of deceiving the Rx irrespective of which strategy he chooses. If he intercepts the Tx's communication (cipher), he could, even though he cannot interpret the cipher, cause the Rx to act contrary to the Tx's intention by simply substituting the other cipher for the one actually sent by the Tx. Similarly, although he would not know which action the receiver would take since he does not know the key chosen by the Tx and Rx, he could send either cipher, "Buy" or "Sell", with the assurance that it would be accepted and acted on by the Rx. In either event, the opponent would be certain of deceiving the Rx to act in a way not requested by the Tx.

To protect against this sort of deception by outsiders, there is essentially only one strategy available to the Tx and Rx. They must enlarge the set of messages that can be sent through the channel so that for any particular choice of an encoding rule (corresponding to a choice of a key for the secrecy channel) there will be some messages that will be acceptable to the Rx, i.e., that the Tx might send to the Rx according to the encoding rule (protocol), while others would be rejected as unauthentic since the Tx would not have used them (under the chosen encoding rule). In other words, the opponent must be uncertain of which messages will be acceptable to the receiver in all cases. Message authentication is critically dependent on this uncertainty, and on how it is distributed over encoding rules, messages and source states. Ideally, in analogy to perfect encryption schemes, this should be done in such a way that an opponent has no better chance of deceiving the Rx if he waits and observes a legitimate message than he would have had, had no observation been made. Again, in the smallest possible example, i.e., of a one-bit source, there are two equally likely instructions, say buy and sell as before. Instead of two messages however, we shall now use four, which requires that two bits actually be communicated. These two bits, if used optimally will inform the Rx of the Tx's intention (one bit) and provide precisely one bit of authentication; i.e., the opponent's probability of deceiving the Rx will be 1/2 irrespective of whether he chooses to impersonate the Tx or to wait and observe a legitimate message and then substitute some other message in its stead.

Probably the most commonly encountered authentication scheme in practice is one that uses an authenticator appended to the Tx's intended communication, say Hi or Lo appended to the instruction to buy or sell for this example. The four messages in this case would then be Buy-Hi, Buy-Lo, Sell-Hi and Sell-Lo where the first part of the message is the (unencrypted in this example) instruction and the second part is the appended authenticator. In the example there are also four encoding rules (corresponding to keys in the encryption example), a particular one of which is chosen with uniform probability distribution by the Tx and Rx flipping a fair coin twice in advance of their needing to authenticate a message (and in secret from the opponent, denoted by the labels HH, HT, TH and TT. The specific authentication protocol we will discuss, out of several possible protocols satisfying the conditions of this example, is the Cartesian product construction

$$\begin{array}{c}H\\T\end{array}\begin{bmatrix}\text{Buy} & - \\ - & \text{Buy}\end{bmatrix} \otimes \begin{array}{c}H\\T\end{array}\begin{bmatrix}\text{Sell} & - \\ - & \text{Sell}\end{bmatrix} = \begin{array}{c|cccc|} & \text{Buy-Hi} & \text{Buy-Lo} & \text{Sell-Hi} & \text{Sell-Lo}\\ \hline \text{HH} & \text{Buy} & & \text{Sell} & \\ \text{TT} & \text{Buy} & & & \text{Sell} \\ \text{HT} & & \text{Buy} & \text{Sell} & \\ \text{TH} & & \text{Buy} & & \text{Sell} \\ \hline \end{array}$$

If the opponent knowing the protocol shown above but not the particular encoding rule chosen by the Tx and Rx, attempts to impersonate the Tx and send a message to the Rx, since there are four equally likely encoding rules, for each of which there are only two acceptable (to the Rx) messages, there is clearly only a 50-50 chance that the message the opponent chooses will be one of the two acceptable ones for the particular encoding rule being used by the Tx and Rx. If, on the other hand, the opponent waits to observe a message sent by the Tx, his uncertainty about the encoding rule they are using will have shrunk from one in four equally likely possibilities to one of two. However, there is a single, but different, acceptable substitute message in each case depending on which rule is being used by the Tx and Rx. For example, if the observed message is Buy-Hi, then the encoding rule must be one of the pair labeled HH and HT. In the first case Sell-Hi would be accepted by the receiver and Sell-Lo would be rejected as unauthentic, while the reverse would be true if the chosen encoding rule was the one labeled HT. This example illustrates a two-bit authentication scheme for which equality holds in (1) that communicates one bit of information and provides one bit of authentication capability at the expense of transmitting two bits of information in each message. The important point to this example is that the confidence the transmitter and receiver can have in the authentication of their communication is provable, i.e., independent of the computational capabilities of an opponent. Brickell [12], Stinson [13,14], Simmons [7,8,9,10] and others have devised various provably secure authentication schemes or codes. The bound in (1) applies to all of these codes, many of which are also perfect in the sense that equality holds. These codes make it possible to provide arbitrarily high levels of confidence in the authenticity of messages, at the expense of a very large usage of key-like secret information defining the selection of encoding rules to be used.

For many real-world applications, the authentication "key" distribution problem described above is avoided by using cryptographic concealment of the sets of acceptable and unacceptable messages. Considering again the one-bit source example -- the outcome of a fair-coin toss in this case -- used earlier, the transmitter and receiver could use standard encryption techniques to generate the sets of messages which the transmitter will send and the receiver will accept. With no loss of generality, it is assumed that it is public knowledge that the transmitter and receiver have agreed that the 64-bit binary sequence 11...1 will denote the outcome "heads" and the string 01...1 will denote "tails". In other words, the redundant information used to authenticate a message is the suffix of 63 1's and only the left-most bit in a sequence conveys the outcome of the coin toss. To protect themselves from deception by the opponent the transmitter/receiver arrange (in advance) to encrypt whichever of these sequences the coin toss indicates using the data encryption standard (DES) cryptoalgorithm and a secret (known only to them) DES key, which, as is well known, consists of 56 bits of equivocation to an outsider, the wiretapper. Each of the 2^{56} possible choices of a DES key corresponds in this scheme to a choice of an authentication encoding rule. Consequently, $|\mathcal{E}| = 2^{56}$, and the bound (1) says that even if the transmitter/receiver choose among the 2^{56} encoding rules optimally, they cannot limit the opponent's probability of successfully deceiving the receiver into accepting an unauthentic message to less than

(2) $$P_d \geq \frac{1}{\sqrt{|\mathcal{E}|}} = \frac{1}{2^{28}} \approx 3.7 \times 10^{-9}$$

or roughly four parts in a billion.

In practice, the opponent's chances of success are dramatically less than (2) would suggest. There are 2^{64} possible ciphers (messages), only two of which are acceptable for any particular choice of a key (authentication encoding rule). Therefore, if the opponent merely selects a cipher at random and attempts to impersonate the transmitter, his probability of success is 2^{-63} or approximately one chance in 10^{19}. The question is, can he do better. As far as impersonating the transmitter is concerned, the answer is essentially no, even if he has unlimited computing power. For each choice of an encoding rule, there are two (out of 2^{64}) ciphers that will be acceptable as authentic. Assuming that the mapping of the sequences 11...1 and 01...1 into 64-bit cipher sequences under DES keys is a random process, this says that the total expected number of acceptable ciphers (over all 2^{56} keys) is $\approx 2^{56.9888}$, i.e., ϵ close to 2^{57}. Even if the opponent could feasibly carry out the enormous amount of computation that would be required to permit him to restrict himself to choosing a cipher from among this collection, his chances of having a fraudulent message be accepted by the receiver would still only be $\approx 2^{-56}$ or roughly one chance in 10^{17} which is what we meant when we said that the answer was essentially no since 10^{-17} isn't much different than 10^{-19} while both differ enormously from the bound, (2), of $\approx 10^{-9}$. The opponent could not do better, nor worse, (in attempting to impersonate the transmitter) even if he possessed infinite computing power than choose a cipher randomly, with a probability distribution weighted to reflect the number of times each cipher occurs, from among the $\approx 2^{57}$ potentially acceptable ciphers.

However, the channel bound in (1) applies to all authentication schemes, hence the apparent contradiction must arise in connection with the opponent's substitution strategy. If the opponent waits to observe a legitimate message (cipher), can the information acquired by virtue of this observation be put to practical use to improve his chances of deceiving the receiver? Even if he doesn't know the state of the source, he knows that the cipher is the result of encrypting one of the two 64-bit sequences 111...1 or 011...1 with one of the 2^{56} DES keys. He also knows that with a probability of essentially one (≈ 0.996), there is only one key that maps the observed cipher into either of these two sequences, hence, he is faced with a classical "meet in the middle" cryptanalysis of DES. Clearly if he succeeds in identifying the DES key, i.e., the encoding rule being employed by the transmitter receiver, he can encrypt the other binary string and be certain of having it be accepted, and hence be certain of deceiving the receiver. The point, though, is that in order for him to make use of his observation of a message he must be able to determine the DES key the transmitter and receiver(s) are using, i.e., he must be able to cryptanalyze DES. If he can do this, the expected probability of deceiving the receiver is ϵ close to one, the small deviation being attributable to the exceedingly small chance that two (or more) DES keys might have encoded source states into the same message (cipher). Thus, we have the paradoxical result that the practical system is some eight or nine orders of magnitude more secure than the theoretical limit simply because it is computationally infeasible for the opponent to carry out in practice what he should be able to do in principle. In this respect, practical message authentication is closely akin to practical cryptography where security is equated to the computational infeasibility of inverting from arbitrarily much known matching cipher text and plaintext pairs to solve for the unknown key, even though in principle there is more than enough information available to insure a unique solution.

The example of the preceding paragraphs illustrates very clearly the first essential dichotomy in authentication schemes, namely the division according to whether the security of the authentication is provable, i.e., independent of the computing power and time the opponent may bring to bear, or else dependent on the infeasibility of his being able to carry out in practice a computation that in principle he could do:

provably secure -- computationally secure

To demonstrate the essential nature of the next dichotomy in authentication schemes, we need to briefly examine the two most widely used schemes at present. Authentication has traditionally been achieved by way of encryption using single-key cryptoalgorithms since, until recently, this was the only type of cryptography available. In a common U. S. military authentication protocol, for example, both the transmitter and receiver are provided with a matching pair of sealed authenticators distributed with the same physical security with which the cryptographic keys are handled; actually a random sequence of symbols produced and distributed by the National Security Agency. The sealed packets are constructed so as to provide a positive indication (tattle-tale) if they are opened. Each communicant is responsible for the protection of his sealed authenticator and is administratively constrained from opening it until it is to be used. To authenticate a message, the transmitter opens his sealed packet, appends the enclosed authentication suffix to the message and then either block encrypts the resulting extended message or else encrypts it with cipher or text feedback so that the effect of the appended authenticator is spread throughout the resulting cipher under the control of the secret key. This cipher is then transmitted as the authenticated message. The receiver, upon receiving and decrypting the cipher, opens his matching sealed authenticator and accepts the message as genuine if the cipher decrypted to a string of symbols with an authenticating suffix matching his authenticator, and otherwise rejects it as unauthentic.

Because of the sensitivity of the authenticators, i.e., anyone having access to one could authenticate a fraudulent message of his own choice, they must generally be handled under two-man control both in distribution and in the field prior to their use which greatly complicates their distribution and control, and more importantly limits the physical environments in which they can be used. Since the key in a single-key cryptoalgorithm must also be handled in the same way, the sealed authenticators have only marginally affected the physical security requirements, and hence have generally been acceptable for military and diplomatic communications. In addition, and even more significant for the present discussion, the cipher that is the authenticating message must be completely inscrutable to an outsider, otherwise it would be possible to strip an authenticator off an authentic message and attach it to a fraudulent one. The point is that while this classic military authentication scheme does provide secure authentication of information, it does so at the expense of requiring complete secrecy for the information being authenticated.

On the other hand, the authentication of electronic funds transfers in the Federal Reserve System does not require nor result in secrecy for the information being authenticated. By directive of the Secretary of the Treasury [15], all such transfers must be authenticated using a procedure that de facto depends on the Data Encryption Standard (DES) single-key cryptographic algorithm. The protocol includes precise format requirements, etc., however the essential feature for our purposes is that an authenticator is generated using a DES mode of operation known as block chaining. The information to be authenticated is first broken into blocks of 64-bits each. The first block is added modulo two (exclusive or) to a 64-bit initial vector (IV), which is changed daily and kept secret, and the sum encrypted using a secret

DES key (known to both the transmitter and the receiver). The resulting 64-bit cipher is then exclusive or'ed with the second block of text and the result encrypted to give a second 64-bit cipher, etc. This procedure is iterated until all blocks of the text have been processed. The final 64-bit cipher is clearly a function of the secret key, the initial vector, and of every bit of the text, irrespective of its length. This cipher is appended as an authenticator to the information being authenticated to form an extended message which is normally transmitted over an open communications channel, although it may be superencrypted if privacy is desired. However, this operation is independent of the authentication function. The authenticator can be easily verified by anyone in possession of the secret key and the initial vector by simply repeating the procedure used by the transmitter to generate it in the first place. An outsider, however, cannot generate an acceptable authenticator to accompany a fraudulent message, nor can he separate an authenticator from a legitimate message to use with an altered or forged message since the probability of it being acceptable in either case is the same as his chance of "guessing" an acceptable authenticator, i.e., one in 2^{64}. In this authentication scheme, which is a classic example of an appended authenticator, the authenticator is a complex function of the information that it authenticates, as well as the secret key and initial vector.

The important point which this example illustrates is that unlike the classic military authentication scheme where secrecy was an essential part of being able to authenticate information, the EFT authentication scheme does not require that the information being authenticated be kept secret. There is no particular reason for it not being secret, and as was indicated there are instances in which the extended message may be superencrypted (independent of authentication) to provide privacy but the authentication procedure does not itself conceal the information. Because of the intrinsic nature of single-key cryptography, however, the appended authenticator in the extended message is necessarily inscrutable to an outsider not in possession of the key, since anyone possessing the secret key and initial vector is, in addition to being able to verify the appended authenticator to a legitimate message, also able to authenticate an arbitrary fraudulent message.

In the early 70's, Sandia encountered the problem of authenticating data from seismic stations that had been designed to verify compliance (by the Russians) with a proposed comprehensive nuclear weapons test ban treaty [16]. Secrecy was not possible in this application since the Russians had to be able to "see" the information being communicated in order for the scheme to be acceptable to them, otherwise the U. S. could have conceivably transmitted information other than what had been agreed to by treaty. On the other hand, in order for the system to be acceptable to the Americans, it had to be true that the Russians, even though they could examine the authenticated message and verify the information it contained, not be able to utter a fraudulent message which the U. S. would accept as authentic. Apparently this application was the first in which authentication without secrecy was required.

Recall that the first discussion of two-key (read also public-key) cryptography in the open literature occurred several years later so that the only authentication schemes available for a system that was to be shared with the Russians in 1972 had to depend on conventional single-key cryptographic techniques, applied so as to approximate the desired end of authentication (to the monitor) without secrecy (to the host) in a scheme very similar to the EFT scheme described above. The compromise solution, found by Simmons, Stewart and Stokes in 1972 [17], was to form an authenticator that was much shorter than the message, where the authenticator was made to be a function of the entire message by repeated encryptions of blocks of text operated on by intermediate ciphers. The authenticator was then block encrypted and appended to the unencrypted message. This authentication protocol is currently referred to as a message authentication code (MAC) [18], and has a much cleaner implementation, for example, in the EFT authentication protocol. This solved the problem of making it possible for the host to monitor the seismic data in real time as it was transmitted, however, the encrypted authenticator was still inscrutable until he was later given the key with which it had been encrypted. Ironically, the host and monitor each trusted the resulting system to the same level of confidence for the same reason. The monitor trusts the authentication since in order to create a forgery the host would have to invert from a known plaintext/cipher pair, i.e., break the cryptosystem by cryptanalysis, to find the key used by the monitor. On the other hand, the host is satisfied that the monitor didn't conceal information in the preceding transmission if the key he is given generates the authenticator that was transmitted since the monitor would have had to solve for the (unique?) key relating the plaintext and a desired bogus authenticator that concealed a hidden message if he wished to cheat; i.e., the monitor would have to solve precisely the same hard problem on which he bases his confidence in the authenticator.

To shorten the periods of implicit trust required of the host, keys can be generated sequentially by the same cryptoalgorithm used to encrypt the authenticator so that for all intents there is an unlimited number of session keys available. This makes it feasible to process shorter blocks of data using a unique session key for each block, with a flow of session keys being made available to the host after essentially only the delay of a two-way satellite relay link. In the limit, with block size and the two-way delay, such a scheme is asymptotic to a true message authentication without secrecy system, although there is a period in each iteration during which the host is at greater risk of being cheated than is the monitor.

The discussion of the preceding paragraphs defines the second essential dichotomy in authentication schemes, namely whether the authentication is carried out with or without secrecy. Using single-key cryptographic techniques, it does not appear to be possible to completely realize the goal of having no uncertainty (secrecy) in the authenticated message, although we have described a procedure that gets asymptotically close to this objective. As we shall see in the next section, by using two-key cryptographic techniques it is possible to realize authentication without secrecy, with no compromise required. The second dichotomy to authentication though is:

with secrecy -- without secrecy

In all of the authentication schemes discussed thus far, since the transmitter and receiver must both know the same secret (from the opponent) information (either the key in a simple key cryptographic algorithm, or a sealed authenticator and a cryptographic key or an initial vector and a cryptographic key, etc.) they can each do anything the other can do. In particular, because of this duality, the receiver cannot "prove" to a third party that a message he claims to have received from the transmitter was indeed sent by the transmitter, since he (the receiver) has the capability to utter an indetectable forgery, i.e., the transmitter can disavow a message that he actually sent. Similarly, the receiver can claim to have received a message when none was sent, i.e., to falsely attribute a message to the transmitter, who cannot prove that he didn't send the message since he could have. In the classic military authentication scheme this is an acceptable situation, since a superior commander doesn't worry that a subordinate will attribute an order to him that he didn't issue and the subordinate doesn't worry that his superior will disavow an order that he did send. There is, in fact, some rudimentary protection against this sort of cheating provided by the sealed authenticators the military uses since if either party can produce his unopened authenticator, it is prima facie evidence that he doesn't know its contents and hence could not have authenticated a message using its contents. In many situations, and in almost all commercial and business applications, the primary concern is with insider cheating, i.e., the person withdrawing cash from an ATM may not be the account holder or the amount shown on a properly signed and valid check may be altered to a larger figure, etc. In a more general model of message authentication, there are four participants instead of three. As before, there is a transmitter who observes an information source and wishes to communicate these observations to a remotely located receiver over a publicly exposed, noiseless, communications channel; a receiver who wishes to not only learn the state of the source (as observed by the transmitter) but also to assure himself that the communications (messages) he accepts actually were sent by the transmitter and that no alterations have been made to them subsequent to the transmitter having sent them and an who opponent wishes to deceive the receiver into accepting a message that will misinform him as to the state of the source. In addition, there is a fourth party, the arbiter. The arbiter's sole function is to certify on demand whether a particular message presented to him is authentic or not, i.e., whether it is a message that the transmitter could have sent under the established protocol. He can never say that the transmitter did send the message, although the probability that it could have come from a source other than the authorized transmitter can be made as small as one likes, only that he could have under the established protocol.

Since we wish to use the problem of authenticating seismic data to verify compliance with a comprehensive nuclear weapons test ban treaty to illustrate the third dichotomy, we return briefly to the problem of achieving true message authentication without secrecy. Two-key (nee public key) cryptography provides an immediate solution to this problem since the essential property of two-key cryptography is the separation of the secrecy channel from the authentication channel, which are inextricably linked in single-key cryptosystems. In two-key cryptography, the encrypt and decrypt keys are not only different, but it is also computationally infeasible to determine at least one of the keys from a knowledge of the other key and of arbitrarily many matched plaintext message/cipher pairs. If the receiver (decrypt) key cannot be deduced from a knowledge of the transmitter (encrypt) key, then the transmitter key may be publicly exposed, so long as the receiver key is kept secret, without jeopardizing the transmitter's ability to communicate in secret to the receiver, although the receiver cannot authenticate the source of the communication, i.e., cannot be sure of the origin of the ciphers. This is the secrecy channel. Conversely, if the transmitter's encrypt key cannot be recovered from a knowledge of the receiver's decrypt key, etc., then, although secrecy is impossible, the receiver can be confident that the communication originated with the purported transmitter and that the message has not been altered in transit if the transmitter can be unconditionally trusted to keep the encrypt key secret. This is the authentication channel.

The obvious solution to the authentication without secrecy problem exploits the authentication channel. The U. S. would install a two-key cryptographic system along with the seismic sensor package in the borehole with a secret (known only to the U. S.) encrypt key that would be volatilized if the package was tampered with. The decrypt key would be shared with the Russians (and perhaps with third parties who need to be able to authenticate transmissions). The messages are the ciphers obtained by encrypting the seismic data along with agreed upon identifiers -- station ID number, date, clock, message number, etc., -- which are required, not only for their obvious utility, but also to provide the redundant information needed by the U. S. to authenticate the messages. The Russians could decrypt the ciphers in real time, perhaps even delaying the transmission in a data buffer for the time required to decrypt, to satisfy themselves that nothing other than the agreed upon seismic data and prearranged formatting information were present. Thus no part of the transmission would have to be kept secret from the Russians. Similarly, the U. S. would decrypt the cipher upon receipt and accept the transmission as authentic if and only if the expected redundant formatting information was present. This scheme depends only on the availability of an authentication channel, separate from the secrecy channel, and hence is not dependent on any particular two-key cryptoalgorithm.

Unfortunately, although the system just described allows the monitor to authenticate messages to whatever level of confidence he requires while at the same time permitting the host to reassure himself that no unauthorized information is concealed, it does not permit arbitration of disputes between the host and the monitor. If unilateral response by the monitor, such as abrogation of a treaty or resumption of atmospheric testing of nuclear weapons as the U. S. did in 1962 in response to the Soviets 1961 violation of the Joint Understanding of a moratorium on such tests, is the only action to result from a detection by the monitor of a violation of the agreement, the compromise system just described suffices since the monitor can be unilaterally convinced of the authenticity of the seismic information that indicates a violation of the treaty. If, however, the action to be taken by the monitor in the event that a violation is detected involves convincing third parties or arbiters, such as the United Nations, NATO, etc., then it must be impossible for the monitor to forge messages. Otherwise, the host could disavow an incriminating message as being a forgery fabricated by the monitor (i.e., disinformation in the current Washington terminology), an assertion which the monitor can not disprove since he has the known capability to encrypt messages and hence to create undetectable forgeries. The problem is that an acceptable (to the monitor) authentication scheme for this application must also include a capability to logically prove the authenticity of information to impartial third parties.

Various cooperative schemes were considered in which each of the three legitimate participants, the transmitter, the receiver and the arbiter(s), contributed to the key in a two-key algorithm in such a way that the resulting combined key was totally uncertain to each of them; for example, the exclusive or of three independent keys. Practical difficulties having to do with trust in equipment as opposed to the logical soundness of the protocol, finally forced the concatenation of three (or more if there is more than one arbiter) separate and independent two-key authentication channels. Each user supplies authentication equipment of his own construction operating with an encryption key that only he knows. His equipment operates on data and cipher streams from other equipments and communicates cipher streams to the other equipments and to the instrumentation cab at the top of the borehole. The decryption keys are shared by all parties. While it is true that each party can perform any operations that his -- supposedly secret -- downhole equipment can, this doesn't make it possible for him to utter an acceptable forgery since the other cryptosystems are inscrutable to him since he does not know the (secret) encryption keys they are using. Furthermore, any party by publicizing the secret information he is supposed to protect could only make it possible for the other parties to duplicate the actions of two out of the three or more encryption systems. This concatenated encryption system renders it impossible for the host to disavow incriminating messages by unilaterally compromising his key. From the monitor's standpoint, even if the host and the arbiter collude to deceive him, he will still be able to establish, to his satisfaction, the authenticity of messages. In the improbable event that all of the other parties conspire to defraud the monitor, the monitor will still know whether a message is authentic or not but will be unable to persuade impartial (and uninvolved) observers that he is telling the truth. In other words, the worst that could happen, from the monitor's standpoint, for this authentication scheme is that he could, with low probability, find himself in the same situation that he was faced with with certainty in the system described earlier. It should be noted that the three participants need not use the same cryptoalgorithm, nor the same key sizes, etc. All that is required is that each provide a two-key authentication channel and share their decryption key with all other participants.

The resulting system provides authentication without secrecy with the capability to arbitrate host (transmitter) and monitor (receiver) disputes in an authentication scheme that is only computationally secure, i.e., the security is no better than the concealing cryptoalgorithm is secure. The essential dichotomy in authentication schemes illustrated by this application hinges on the question of whether the transmitter and receiver trust each other unconditionally or not. We choose to characterize this division by terminology that suggests the consequences of a failure of this trust.

with arbitration -- without arbitration

Simmons has recently constructed several classes of provably secure authentication codes that permit arbitration of transmitter/receiver disputes called $\mathbf{A}^2$ codes [19,20]. The relationship between these $\mathbf{A}^2$ codes and the computationally secure cryptographic based authentication with arbitration schemes just described is almost exactly the same as the relationship of the provable secure authentication codes, **A** codes, described earlier and the related computationally secure cryptographic based authentication schemes. For the sake of completeness, we exhibit a provably secure authentication with arbitration code for a one-bit source that is the extension of the Cartesian product construction of an authentication code described earlier. Clearly, in any authentication scheme there must be some uncertainty as to what messages will be accepted and/or certified by the arbiter for any participant whose cheating is to be prevented. In particular, for $\mathbf{A}^2$ codes this means that the opponent (outsider) must be uncertain as to which messages the receiver will accept (the opponent doesn't care whether the arbiter will later certify an accepted message as having come from the transmitter or not); however, the receiver (insider) must be uncertain as to which messages the arbiter will certify and the transmitter (insider) must also be uncertain as to which messages the receiver will accept. It turns out that there are infinite families of $\mathbf{A}^2$ codes in which all of these uncertainties to the various potential cheaters can be made to be the same and hence in which the resulting codes are perfect in the sense of channel usage described earlier. Our example is the smallest of these codes possible -- one bit of information is communicated in the message and one bit of uncertainty is presented to insiders and outsiders alike. Consider the Cartesian construction for authenticating rules:

$$A = \begin{pmatrix} H & H & - & - \\ H & - & H & - \\ - & H & - & H \\ - & - & H & H \end{pmatrix} \otimes \begin{pmatrix} T & T & - & - \\ T & - & T & - \\ - & T & - & T \\ - & - & T & T \end{pmatrix}$$

or

$$A = \begin{array}{c|cccccccc|} & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 \\ \hline a_1 & H & H & - & - & T & T & - & - \\ a_2 & H & H & - & - & T & - & T & - \\ & \vdots & & & & \vdots & & & \vdots \\ a_{15} & - & - & H & H & - & T & - & T \\ a_{16} & - & - & H & H & - & - & T & T \\ \hline \end{array}$$

where the source is a toss of a fair coin, H or T. The authentication with arbitration protocol calls for the receiver to choose one of the authenticating rules, a_i, with a uniform probability distribution. For example, a_1, says that a head outcome to the transmitter's coin toss could be communicated by the transmitter using either message m_1 or m_2. Similarly, messages m_5 or m_6 would communicate source state "tails", while messages m_3, m_4, m_7 and m_8 would be rejected by the receiver under rule a_1 as unauthentic. The important point to note is that in each of the authenticating rules there are exactly two acceptable (to the receiver) messages available for each state of the source. The receiver communicates his choice of an authenticating rule to the arbiter in secret (from the transmitter and the opponent(s)). According to the protocol, the receiver has committed himself to accepting as authentic precisely those four messages corresponding to the source states in the authenticating rule he chose and to rejecting the remaining four as unauthentic. The arbiter randomly chooses one of the messages from the pair that would communicate "heads" and one from the pair that would communicate "tails" to form an encoding rule which he then forwards (in secret from the opponent and the receiver) to the transmitter. In this particular example for each choice of an authenticating rule there are four possible encoding rules, one of which is chosen by the arbiter with a uniform probability distribution. It is also the case that each possible encoding rule occurs in four different, but equally likely, authenticating rules so that the transmitter is uncertain as to the authenticating rule chosen by the receiver even though he knows two messages (one communicating source state "heads" and one communicating source state "tails") used in that rule. For example, assume that the receiver chose a_1, and that the arbiter constructed the encoding rule:

$$\begin{array}{c|cccccccc} & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 \\ \hline e & - & H & - & - & T & - & - & - \end{array} .$$

Source state H would be communicated under the established protocol by the transmitter sending message m_2 while T would be communicated by sending message m_5. According to the protocol, m_2 and m_5 will not only be accepted as "authentic" by the receiver, but will also be certified by the arbiter as messages the transmitter could have sent. Of course, the receiver would also accept m_1 and m_6 as authentic, however the arbiter would not certify either of these messages as ones the transmitter could have sent under the established protocol.

Using this authentication scheme we now show that the immunity provided to each of the five types of cheating described earlier is to hold the cheater to a probability of 1/2, i.e., one bit of protection, as claimed irrespective of which type of cheating is considered. The easiest of the deceptions to analyze is the case of the outsider (opponent) who only knows the "system", i.e., he knows what the procedures are but does not know the receiver's or arbiter's choices. It should be clear that if he attempts to impersonate the transmitter and send a message when none has been sent, his probability of choosing one of the four (out of eight) messages that the receiver has agreed to accept (in his choice of an encoding rule) is 1/2 since in each case there are four equally likely messages that will be accepted as authentic and four that will be rejected as unauthentic. On the other hand, if he waits to observe a message, say m_1, his uncertainty about the encoding rule chosen by the receiver drops from one out of 16 equally likely candidates to one out of four, however these four leave him with four equally likely possibilities for the message that the transmitter is to use to communicate the other state of the source, and much more importantly, with four equally likely pairings of messages that the receiver would accept as communicating the other state of the source, with each message occurring in precisely two of the pairs. The net result is that the opponent's probability of success in substituting a message that the receiver will accept as communicating the other state of the source is still 1/2.

Consider next the case of the transmitter disavowing a message that he actually sent. In order to succeed, the transmitter must not only choose a message that the receiver will accept, but also one that is not used in the encoding rule forwarded by the arbiter. In other words he must choose a message that was used in the authenticating rule that the receiver chose, but not used in the encoding rule generated by the arbiter's choice. Continuing with the example used above, the transmitter knows from the encoding rule that was given to him by the arbiter that the receiver must have chosen one of the four authenticating rules:

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
H	H	-	-	T	T	-	-
H	H	-	-	T	-	T	-
-	H	-	H	T	T	-	-
-	H	-	H	T	-	T	-

Since messages m_3 and m_8 do not appear in any of these rules, the transmitter can be certain that either of these messages would be rejected by the receiver as unauthentic. Each of the remaining four messages, m_1, m_4, m_6, and m_7, appears in two of the equally likely choices of an authenticating rule, hence he cannot do better than choose one out of these four messages with equiprobability. Irrespective of which of the four he chooses, the probability that it will be accepted by the receiver is 1/2. If it is accepted, the transmitter can disavow having sent it, since he knows that the arbiter will not certify it as a message that would have been used under the established protocol.

Finally, we consider the two types of cheating available to the receiver. Of the four messages that he has agreed (with the arbiter) that he will accept as authentic, since they are used in his choice of an authenticating rule, two will be certified as being messages that could have been used under the established protocol and two will not be certified. The receiver will succeed in fraudulently attributing a message to the transmitter if he is able to choose one of the pair that the arbiter will certify and will fail otherwise. It should be clear that his probability of success is 1/2 since the arbiter's selection procedure chooses among the acceptable (to the receiver) messages with a uniform probability distribution. If he waits until he receives a message from the transmitter communicating a source state, say m_2, indicating that the outcome of the coin flip was heads, he is still totally uncertain as to which of the messages that could be used to communicate the source state "tails" will be certified by the arbiter. The result however is that either message m_5 or m_6 is equally likely to be the one that will be certified, and his probability of successfully substituting a message conveying a different state of the source than was communicated in the message sent by the transmitter, i.e., of substituting one which will both communicate a different state of the source and will subsequently be certified by the arbiter as a message the transmitter could have sent under the established authentication protocol is 1/2.

This example illustrates all of the essential features of authentication codes that permit arbitration, A^2-codes. Three bits of information had to be communicated to specify one of eight equally likely messages. According to the protocol, this communication provides one bit of information (to the receiver) about the source state, one bit of protection (to the transmitter and receiver) against deception by outsiders and one bit of protection (to the transmitter or to the receiver) against cheating by insiders. Since the probability of success for the "cheater" in all cases is simply the probability that a randomly chosen message will be successful (in cheating) it seems reasonable to describe the code illustrated here as perfect. As has been pointed out in earlier papers on authentication codes without arbitration, these "perfect" codes are also perfect in the natural sense that all of the information transmitted is used either to communicate the state of the source or else to confound one of the cheating parties.

Conclusion

The purpose of the lengthy description of the various authentication problems and schemes given here was two-fold: first to persuade the reader that each of the three essential dichotomies is genuine and arises in real-world situations and secondly, by construction of solutions to show that these really are independent classifications of authentication schemes. Figure 1 summarizes the resulting classification and provides one or more examples of schemes illustrating each basic class. For the more recent contributions to authentication theory, the principal authors and/or collaborating coauthors are also indicated.

	with secrecy	without secrecy
with arbitration	digital signatures (concatenated encryptions) Rivest, Shamir, Adleman	encryption with secret key Rivest, Shamir, Adleman; 3-party two-key cryptoalgorithm treaty verification scheme Simmons; Brickell, DeLaurentis
without arbitration	classic military authentication scheme	message authentication code; Federal Reserve EFT authentication scheme; single-key cryptoalgorithm treaty verification scheme Simmons, Stewart, Stokes

computationally secure

	with secrecy	without secrecy
with arbitration	replicated Cartesian A^2-codes (far from perfect); perfect (?)	Cartesian A^2-codes Simmons
without arbitration	pairwise balanced non-Cartesian A-codes Brickell; Simmons; Stinson	Cartesian A-codes Gilbert, McWilliams, Sloan; Brickell; Simmons; Stinson

provably secure

Figure 1. The Natural Taxonomy of Authentication Schemes

References

1. D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," Communications of the ACM, Vol. 28, Oct. 1985, pp. 1030-1044.

2. A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," Presented at Crypto'86, Santa Barbara, CA, Aug. 11-15, 1986, pp. 18-1 thru 18-7 of the Conference Abstracts and Papers.

3. O. Goldreich, S. Micali and A. Wigderson, "Proofs that yield nothing but their validity and a methodology of cryptographic protocol design," In The Computer Society of IEEE, 27th Annual Symp. on Foundations of Computer Science (FOCS), pp. 174-187, IEEE Computer Society Press (1986). Toronto, Ontario, Canada, Oct. 27-29, 1986.

4. P. D. Merillat, "Secure stand-alone positive personnel identity verification system (SSA-PPIV)," Sandia National Laboratories Tech. Rpt. SAND79-0070 (March).

5. G. J. Simmons, "A system for verifying user identity and authorization at the point-of-sale or access," Cryptologia, Vol. 8, No. 1, January 1984, pp. 1-21.

6. C. L. Henderson, A. M. Fine," Motion, intrusion and tamper detection for surveillance and containment," Sandia National Laboratories Tech. Rpt. SAND79-0792 (March 1980).

7. G. J. Simmons," A game theory model of digital message authentication," Congressus Numerantium 34 (1982), pp. 413-424.

8. G. J. Simmons, "Message authentication: A game on hypergraphs," Proceedings of the 15th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, March 5-8, 1984, pp. 161-192.

9. G. J. Simmons, "Authentication theory/coding theory," Proceedings of Crypto'84, Santa Barbara, CA, August 19-22, 1984, in Advances in Cryptology, Ed. by R. Blakley, Springer-Verlag, Berlin (1985), pp. 411-432.

10. G. J. Simmons, "The practice of authentication," Proceedings of Eurocrypt'85, Linz, Austria, April 9-11, 1985, in Advances in Cryptology, ed. by Franz Pichler, Springer-Verlag, Berlin (1986), pp. 261-272.

11. E. N. Gilbert, F. J. MacWilliams, N.J.A. Sloane, "Codes which Detect Deception," The Bell System Tech. Journal, Vol. 53, No. 3, March 1974, pp. 405-424.

12. E. F. Brickell, "A Few Results in Message Authentication," Proceedings of the 15th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, March 5-8, 1984, Congressus Numerantium, Vol. 43, Dec. 1984, pp. 141-154.

13. D. R. Stinson, "Some Constructions and Bounds for Authentication Codes," presented at Crypto'86, Santa Barbara, CA, Aug. 12-15, 1986, to appear in Journal of Cryptology, 1987.

14. D. R. Stinson, "A Construction for Authentication/secrecy Codes from Certain Combinatorial Designs," to appear in Journal of Cryptology.

15. Dept. of the Treasury Directive, "Electronic funds and securities transfer policy -- message authentication," Aug. 16, 1984, signed by Donald T. Regan, Secretary of the Treasury.

16. G. J. Simmons, "Verification of treaty compliance -- revisited," Proceedings of the IEEE Computer Society 1982 Symposium on Security and Privacy, Oakland, CA, Apr. 25-27 (1983), pp. 61-66.

17. G. J. Simmons, R. E. D. Stewart, P. A. Stokes, "Digital data authenticator," Patent Application SD2654, S42640 (June 30, 1972).

18. C. H. Meyer and S. M. Matyas, Cryptography: A New Dimension in Computer Data Security, John Wiley & Sons, New York (1982).

19. G. J. Simmons, "A Cartesian Product Construction for Authentication Codes that Permit Arbitration," to appear in J. of Cryptology.

20. G. J. Simmons, "Authentication Codes that Permit Arbitration," Proc. of the 18th Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton, FL, Feb. 23-27, 1987.

ANALYZING ENCRYPTION PROTOCOLS USING FORMAL VERIFICATION TECHNIQUES
(Extended Abstract)

Richard A. Kemmerer

Department of Computer Science
University of California
Santa Barbara, CA 93111

Introduction

Much work has been done in the area of analyzing encryption algorithms, such as DES [Dav 81,Bri 85,BMP 86]. A vast amount of work has also been expended on formally verifying communication protocols [IEE 82,STE 82,RW 83,LS 84,Hol 87]. In contrast, very little work has been devoted to the analysis and formal verification of encryption protocols.

In this paper an approach to analyzing encryption protocols using machine aided formal verification techniques is presented. The idea of the approach is to formally specify the components of the cryptographic facility and the associated cryptographic operations. The components are represented as state constants and variables, and the operations are represented as state transitions. The desirable properties that the protocol is to preserve are expressed as state invariants and the theorems that must be proved to guarantee that the system satisfies the invariants are automatically generated by the verification system.

This approach does not attempt to prove anything about the strength of the encryption algorithms being used. On the contrary, it may assume that the obvious desirable properties, such as that no key will coincidentally decrypt text encrypted using a different key, hold for the encryption scheme being employed.

The following section presents a brief overview of the formal specification language that is used. Next, a sample system is presented along with the desirable cryptographic properties for that system. A formal specification for the example system is then given followed by a discussion about formally verifying and testing encryption protocol specifications. A weakness of the example specification that was discovered through testing the specification is also presented. Finally, a comparison to other work in this area is given, and some thoughts on the usefulness of the approach presented in this paper are discussed.

The Formal Specification Language

Formal specification and verification techniques have become an accepted technique for assuring that a critical system satisfies its requirements. There are a number of formal verification systems available [Rob 79,TE 81,GDS 84,CDL 85,SAM 86]. These systems all use mathematical techniques to guarantee the correctness of the system being designed and implemented. To use these techniques it is necessary to have a formal notation, which is usually an extension of first-order predicate calculus, and a proof theory.

The formal verification system discussed in this paper uses the state machine approach to formal specification. When using the state machine approach a system is viewed as being in various states. One state is differentiated from another by the values of state variables, and the values of these variables can be changed only via well defined state transitions.

The formal specification language that is used in this paper is a variant of Ina Jo*, which is a nonprocedural assertion language that is an extension of first-order predicate calculus. The key elements of the Ina Jo language are types, constants, variables, definitions, initial conditions, criteria, and transforms. A criterion is a conjunction of assertions that specify the critical requirements of a good state. A criterion is often referred to as a state invariant since it must hold for all states including the initial state. An Ina Jo language transform is a state transition function; it specifies what the values of the state variables will be after the state transition relative to their values before the transition. The system being specified can change state only as described by one of the state transforms. A complete description of the Ina Jo language can be found in the Ina Jo Reference Manual [SAM 86].

Before giving a formal specification of the example system a brief discussion of some of the notation is necessary. The following symbols are used for logical operations:

&	Logical AND
$\rightarrow$	Logical implication

In addition there is a conditional form

(if A then B else C)

where A is a predicate and B and C are well-formed terms.
The notation for set operations is:

$\in$	is a member of
$\cup$	set union

* Ina Jo is a trademark of System Development Corporation, a Burroughs Company.

{a,b,...,c} the set consisting of elements a,b,...,and c

{set description} the set described by set description.

The language also contains the following quantifier notation:

$\forall$ for all

$\exists$ there exists

Two other special Ina Jo symbols that may be used are:

N" to indicate the new value of a variable
(eg. N"v1 is new value of variable v1)

T" which defines a subtype of a given type T.

An Example System

The system being used as a pedagogical example in this paper is a single-domain communication system using dynamically generated primary keys and two secret master keys, as described in [MM 80]. The architecture of the system is presented in Figure 1. The terminals communicate directly with the host system, and a new session key (the primary communication key) is dynamically generated by the host for each session. In addition, each terminal has a permanent terminal key that is used by the host to distribute the new session key to a terminal when a new session is initiated. This is the terminal's secondary communication key. Both the terminal keys and the current session keys are stored in encrypted form at the host.

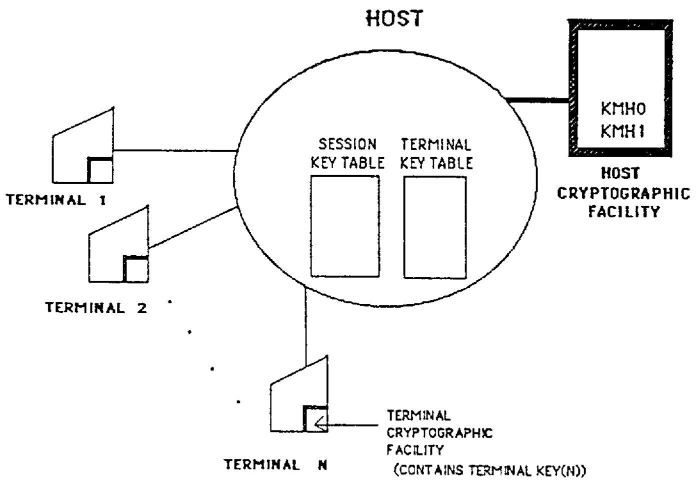

HOST
SESSION KEY TABLE
TERMINAL KEY TABLE
KMH0
KMH1
HOST CRYPTOGRAPHIC FACILITY
TERMINAL 1
TERMINAL 2
TERMINAL N
TERMINAL CRYPTOGRAPHIC FACILITY (CONTAINS TERMINAL KEY(N))

Figure 1 System Architecture

There are two data structures of interest in the host: the terminal key table and the session key table. The ***terminal key table*** is static. Each entry in this table contains the unique terminal key for the corresponding terminal encrypted using a secret master key KMH1. The table looks as follows:

E_{KMH1}(Terminal Key(1))
E_{KMH1}(Terminal Key(2))
:
E_{KMH1}(Terminal Key(i))
:
E_{KMH1}(Terminal Key(n))

In this paper $E_{key-name}$(text) is used to denote text encrypted using the key key-name. Similarly, $D_{key-name}$(text) is used to denote text decrypted using key key-name. Since terminal keys never change, this table is constant for the lifetime of the system.

Unlike the terminal key table, the *session key table* is a dynamic structure. This table is updated each time a new terminal session is started; there is one current session key per terminal. Each entry in the table contains the current session key for the corresponding terminal encrypted using a second secret master key KMH0. The session key table looks as follows:

E_{KMH0}(Session Key(1))
E_{KMH0}(Session Key(2))
:
E_{KMH0}(Session Key(i))
:
E_{KMH0}(Session Key(n))

No terminal key, session key, nor either master key is in the clear in the host. To store the two masters keys a *cryptographic facility* is connected to the host. This facility may be accessed only through the limited cryptographic operations that are provided. In addition, the facility is assumed to be housed in a tamper-sensing container, such as the one described by Simmons [Sim 85], so that the vital information it contains is physically protected. The operations provided by the cryptographic facility are encipher data (ECPH), decipher data (DCPH), and reencipher from master key (RFMK). The interaction between the host and its cryptographic facility is shown in Figure 2.

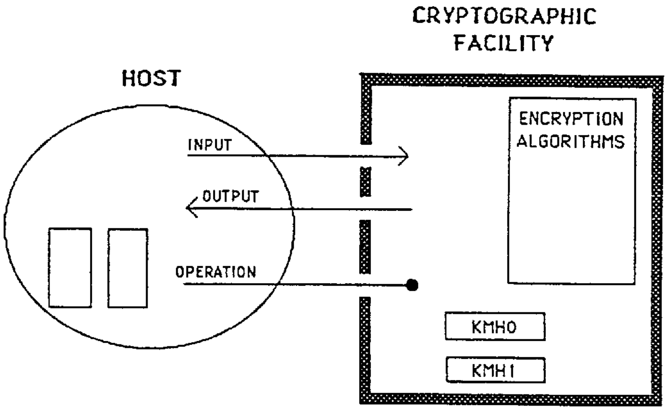


Figure 2 Host and Cryptographic Facility

The *encipher* operation is used when the host wants to send an encrypted message to a terminal. The host provides the clear text message (msg) along with the encrypted form of the appropriate terminal's current session key, E_{KMH0}(Session Key(i)), to the cryptographic facility and is returned the message encrypted using the terminal's session key. Figure 3 illustrates this process.

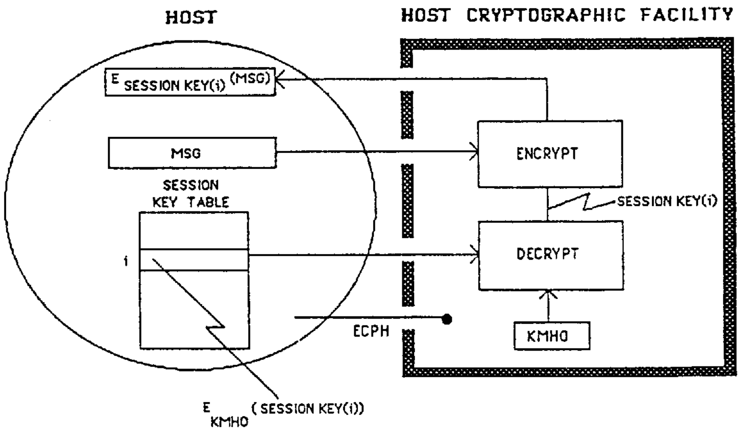


Figure 3 The Encipher Operation (ECPH)

When the host receives an encrypted message from terminal i it uses the *decipher* operation provided by the cryptographic facility in a similar manner to get the message in the clear. That is, the decipher operation must first decrypt the key presented and then use the result to decipher the text presented. Figure 4 illustrates the decipher process.

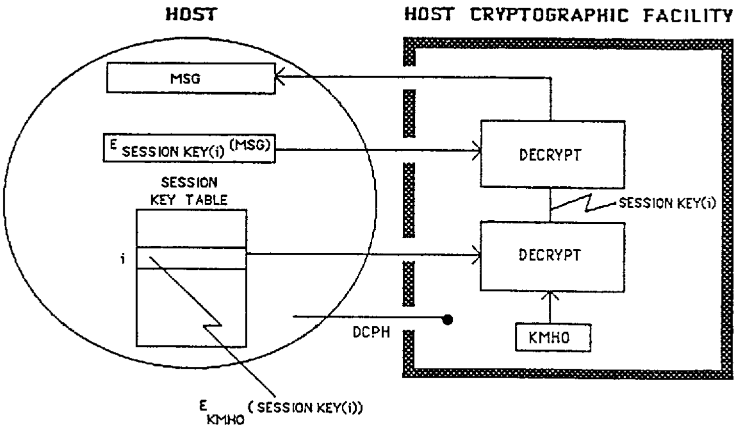


Figure 4 The Decipher Operation (DCPH)

Each time a terminal initiates a session with the host a new session key is needed. Therefore, the host needs access to a *generate session key* operation. It is not necessary, however, to have session key generation be an operation provided by the cryptographic facility. The seemingly contrary requirements of having the host generate the session key and having no key in the clear outside the cryptographic facility are resolved by having the host generate an encrypted version of the session key. The entry in the host's session key table that corresponds to the terminal requesting a session is then replaced by the encrypted key. Meyer and Matyas describe a method of accomplishing this by using a pseudo-random number generator to generate a value that is interpreted as the new session key encrypted using the secret master key KMHO [MM 80].

Since the requesting terminal is also sent a copy of the session key, it is necessary to have an operation to translate the new session key enciphered using the KMHO master key to a form enciphered using the requesting terminal's terminal key. The *reencipher from master key* operation provides this service. It takes two keys as input, decrypts one using KMH1, decrypts the other using KMHO, and then uses the result of the first decryption to encipher the result of the second decryption. Figure 5 illustrates this process.

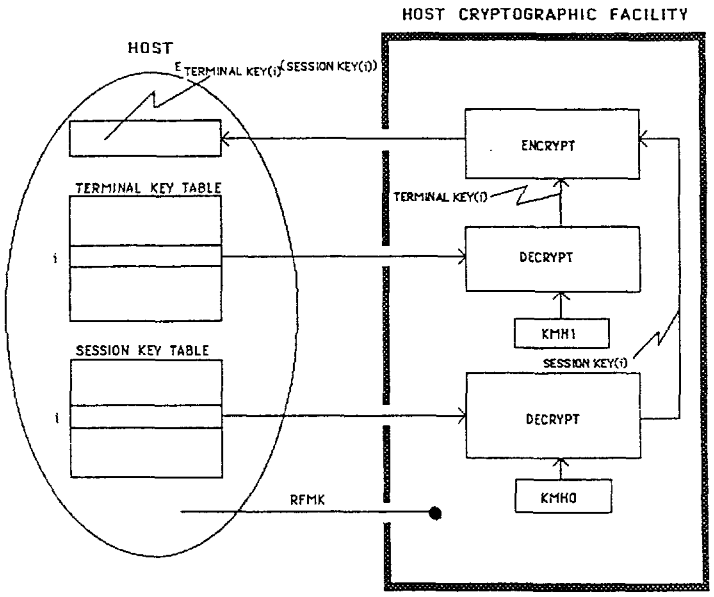


Figure 5 Reencipher From Master Key (RFMK)

In addition to the host's cryptographic facility, each terminal is assumed to have a cryptographic facility that contains its permanent terminal key and that provides operations for encrypting and decrypting messages.

Operations for setting the secret master keys in the host's cryptographic facility or the secret terminal keys in the terminal cryptographic facilities have intentionally been avoided in this paper. It is assumed that these secret keys are distributed by courier or some other trusted means.

An assumption of this system is that the intruder can obtain any information communicated between the host and the terminals. In addition, the intruder can masquerade as an authorized user, and he/she can invoke any of the operations of the host's cryptographic facility.

An obvious desirable property that one may wish to verify about this system is that no clear key exists outside the cryptographic facilities of the host and terminals.

Formal Specification of the Example System

The complete Ina Jo specification for the example system is presented in the appendix. In this section the important aspects of the specification are discussed.

In the example system each terminal has a constant terminal key. This is represented in the model by the Ina Jo constant

Terminal_Key(Terminal_Num): Key.

Similarly, each terminal's session key, which is dynamic, is represented by the Ina Jo variable

Session_Key(Terminal_Num): Key.

As the terms infer, an Ina Jo *constant* is unchanged from state to state and an Ina Jo *variable* may change from state to state. It is the value of the state variables that differentiate one state from another.

The other state variables in the specification are Keys_Used and Intruder_Info, which are both of type information. Keys_Used keeps track of all of the keys that have ever been used by the system. The Intruder_Info variable keeps track of all of the information that the intruder has access to. This includes the contents of the encrypted key tables as well as any information communicated between the host and the terminals.

The four cryptographic operations for the example system are represented by the Ina Jo *transforms* ECPH, DCPH, RFMK, and Generate_Session_Key. The first three correspond to the operations provided by the host's cryptographic facility and the last is provided by the host itself. Since only traffic that is a key or an encrypted form of a key is of interest, there is no need to model any of the send or receive text operations. Also, because it is assumed that the intruder cannot correctly guess random text that corresponds to some encrypted key, the ECPH, DCPH, and RFMK operations change state only when they are invoked with information that the intruder has available. Therefore, each of these operations is written using a conditional form where there is no state change if the information provided is not available to the intruder (i.e., not part of Intruder_Info).

The constants Encrypt and Decrypt represent the encryption and decryption algorithms, respectively. They both take a key and text pair and return text. Properties of the encryption and decryption algorithms are represented in the specification as axioms. An Ina Jo *axiom* is an expression of a property that is assumed. Thus, these qualities are assumed about the algorithms. For instance, one could express the fact that the encryption and decryption functions were commutative by using the axiom

AXIOM $\forall$ t:Text, k1,k2:Key (Encrypt(k1,Decrypt(k2,t))=Decrypt(k2,Encrypt(k1,t)))

Similarly, the following axioms express the properties that no key is an identity function for any text and that no key will correctly decrypt text encrypted using another key.

AXIOM $\forall$ t:Text, k1:Key (Encrypt(k1,t) $\neq$ t)

and

AXIOM $\forall$ t:Text, k1,k2:Key (k1 $\neq$ k2 $\rightarrow$ Decrypt(k2,Encrypt(k1,t)) $\neq$ t)

Note that all of these assumptions are not expressed in the example specification.

The fact that the intruder receives all of the information that is communicated between the host and the terminals is expressed in the Generate_Session_Key transform where the intruder's information is enhanced with the new session key encrypted using the terminal key of the requesting terminal. Also, since the intruder can masquerade as an authorized user, whenever one of the cryptographic operations is invoked the intruder's information is updated with the new information that is produced.

The critical requirements that the system is to satisfy in all states are expressed in the *criterion* clause of the formal specification. For the example system the criterion states that any key that the intruder has (i.e., any key contained in the set Intruder_Info) must not be a key that was used by the system (i.e., a key in the set Keys_Used). Note that this includes keys used in the past as well as those presently being used. The criterion is expressed as follows:

CRITERION $\forall$ k:Key (k $\in$ Intruder_Info $\rightarrow$ k $\notin$ Keys_Used)

The *initial* clause describes the requirements that must be satisfied when the system is initialized. For the example system the initial value of the Keys_Used variable is the set of keys currently being used (i.e., all terminal keys and session keys as well as both master keys). The initial value of the Intruder_Info variable is the appropriate encrypted versions of the terminal and session keys. Because the keys are not required to have any particular value, their values are not specified in the initial clause. However, since the desirable property is for the intruder to never have any keys in the clear, the last conjunct of the initial clause states that none of the encrypted values of the keys can be coincidentally equal to a key being used. That is,

$\forall$ k1,k2:Key (k1 $\in$ Intruder_Info & k2 $\in$ Keys_Used $\rightarrow$ k1 $\neq$ k2).

The need for this additional requirement is discussed in the next section.

To verify that the system specified satisfies the invariant requirements, as expressed in the criteria, two types of theorems are generated. The first states that the initial state satisfies the invariant and the second, which is generated for each transform, states that if the state where the transform is fired satisfies the invariant, then the resultant state will also satisfy the invariant. Thus, by induction all reachable states will satisfy the invariant.

If one can verify the theorems generated, then any system that is consistent with the specification will preserve the invariant. The reader should note that for a system to be consistent with the specification its encryption algorithms must satisfy the axioms stated about encrypt and decrypt.

Formally Verifying the Specification

After the formal specification is completed one can verify the theorems that are generated to check if the critical requirements (Ina Jo criteria) are satisfied. If the theorems are verified and the encryption algorithms satisfy the assumed axioms, then the system will satisfy its critical requirements.

Because the axioms represent the properties that the encryption algorithms are to satisfy, one can verify the system assuming the use of a different encryption scheme by replacing the current axioms with axioms that express the properties of the new encryption scheme.

An advantage of expressing the system using formal notation and attempting to prove properties about the specification is that if the generated theorems cannot be proved the failed proofs often point to weaknesses in the system or to an incompleteness in the specification. That is, they often indicate the additional assumptions required about the encryption algorithm (i.e., missing axioms), weaknesses in the protocols, or missing constraints in the specification. For example, the original specification for the example system did not include the third conjunct that is now in the initial clause. However, without this conjunct the initial clause was not strong enough to imply the invariant. After analyzing the failed proof for some time the possibility of an encrypted version of a key being coincidentally identical to another key was apparent. By adding the third conjunct to the initial clause the problem was avoided. This was a reasonable change to make to the specification since the the occurrence of coincidental values is easy to check when the system is initialized.

Being aware of the coincidental key value problem in the initial clause resulted in a strengthening of the specification for the generate session key operation. That is, the requirements

Encrypt(KMH0,k) ∉ Keys_Used

and

Encrypt(Terminal_Key(Ter),k) ∉ Keys_Used

were added to the formal specification to prevent the encrypted value chosen as a new session key from being coincidentally equal to the value of a key that had been used in the past. This requirement is likely to be harder to realize in an actual system since it requires recording information about all keys that have ever been used.

Testing the Formal Specification

There is a specification execution tool for the Ina Jo language called Inatest [EK 85]. This tool allows Ina Jo specifications to be analyzed by symbolically executing the formal specifications. With the Inatest tool it is possible to interactively introduce assumptions about the system, execute sequences of transforms, and check the results of these executions. This provides the user with a rapid prototype for testing properties of the cryptographic facilities [Kem 85].

Using the Inatest tool revealed the following weakness in the example formal specification. If the secret master keys, KMH0 and KMH1, are equal, then the intruder can obtain a session key in the clear. This flaw is demonstrated by first invoking the Generate_Session_Key transform, which generates a new session key, k. This key is communicated to the requesting terminal (encrypted using the terminal key of the requesting terminal) at the start of the current session; therefore, the encrypted key becomes part of the intruder's information. The DCPH transform is then invoked using the encrypted terminal key from the host's terminal key table and the intercepted session key encrypted using the the requesting terminal's terminal key. Figure 6 illustrates the result of executing the DCPH transform on the two encrypted keys.

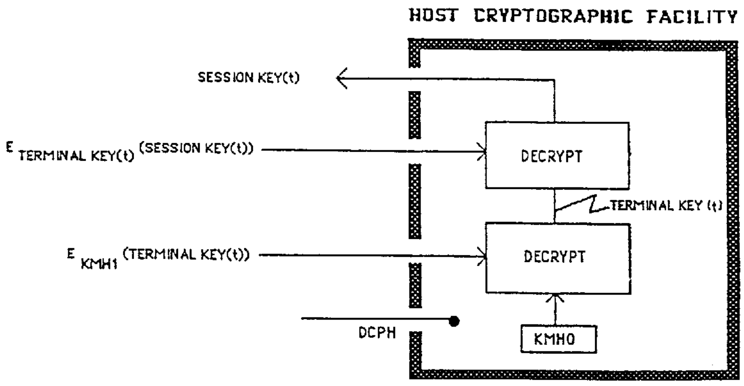


Figure 6 Protocol Flaw Using DCPH

When using the Inatest tool to test a formal Ina Jo specification the user defines a start state, a sequence of transforms (with the appropriate actual parameters) to be executed, and a desired resultant state. To test this weakness the default start state, which is the initial state, was used

Keys_Used=Terminal_Keys $\cup$ Session_Keys $\cup$ {KMH0,KMH1}

& Intruder_Info =

{ks:Key ($\exists$ t:Terminal_Num (ks=Encrypt(KMH0,Session_Key(t))))}

$\cup$ {kt:Key ($\exists$ t:Terminal_Num (kt=Encrypt(KMH1,Terminal_Key(t))))}

& $\forall$ k1,k2:Key (k1 $\in$ Intruder_Info & k2 $\in$ Keys_Used $\rightarrow$ k1$\neq$k2)

The sequence of transforms to be executed consists of the Generate_Session_Key transform followed by the DCPH transform. The parameters of the DCPH transform are the encrypted terminal key for terminal t and the current session key for terminal t encrypted using the terminal key for terminal t. Both keys are known to be part of the intruder's information. The first is from the terminal key table, and the second was sent to terminal t when the current session was started. Letting k represent the key that results from executing the Generate_Session_Key transform on behalf of terminal t, the sequence is,

```
        Generate_Session_Key(t)
```

followed by

```
        DCPH(Encrypt(KMH1,Terminal_Key(t)), Encrypt(Terminal_Key(t),k)),
```

The desired resultant state requires that the key for terminal t that was generated by the Generate_Session_Key transform be part of the intruder's information. This requirement is expressed as:

```
      k ∈ Intruder_Info
```

This is a clear violation of the security requirement since one of the assumptions included in the start state is that k is one of the keys used by the system.

By expanding Generate_Session_Key one gets

```
      ∃ k:Key ∀ t:Terminal_Num (
         Encrypt(KMH0,k) ∉ Keys_Used
       & Encrypt(Terminal_Key(Ter),k) ∉ Keys_Used
       & k ∉ Keys_Used
       & N"Session_Key(t) =
            if t=Ter
               then k
               else Session_Key(t)
       & N"Keys_Used = Keys_Used ∪ {k}
       & N"Intruder_Info = Intruder_Info ∪
               {Encrypt(KMH0,k),Encrypt(Terminal_Key(Ter),k))}
```

The existential is instantiated to k and the resulting information is combined with the start state information.

Next, by expanding the DCPH transform one gets

```
      N"Intruder_Info =
        if  Encrypt(Terminal_Key(t),k) ∈ Intruder_Info
         & Encrypt(KMH1,Terminal_Key(t)) ∈ Intruder_Info
           then Intruder_Info ∪
             {Decrypt(Decrypt(KMH0,Encrypt(KMH1, Terminal_Key(t))),Encrypt(Terminal_Key(t),k))}
           else Intruder_Info.
```

Since both keys are part of the intruder's information this can be reduced to

```
      N"Intruder_Info = Intruder_Info ∪
```

{Decrypt(Decrypt(KMH0,Encrypt(KMH1, Terminal_Key(t))),Encrypt(Terminal_Key(t),k))}.

Since the start state specifies that KMH0=KMH1, KMH1 can be substituted for KMH0 in the innermost Decrypt yielding

N"Intruder_Info = Intruder_Info $\cup$

{Decrypt(Decrypt(KMH1,Encrypt(KMH1, Terminal_Key(t))),Encrypt(Terminal_Key(t),k))}.

Then by applying the first axiom the expression reduces to

N"Intruder_Info = Intruder_Info $\cup$

{Decrypt(Terminal_Key(t),Encrypt(Terminal_Key(t),k))}.

Applying the first axiom again yields

N"Intruder_Info = Intruder_Info $\cup$ {k}.

The desired result follows directly.

This is a well known weakness of using a single master key that is presented in [MM 80]. To strengthen the specification to avoid this particular problem one needs to add the axiom

AXIOM KMH0 $\neq$ KMH1.

Comparison to Previous Work

As was mentioned in the introduction very little work has been devoted to the analysis and formal verification of encryption protocols. In particular, formal verification techniques have not been used in the analysis efforts that have been reported. A notable exception is the Interrogator work of Millen [MCF 87].

The work reported in this paper differs from Millen's work in that the goal of the work being reported is to use existing formal verification tools to formally verify that an encryption protocol specification satisfies its security requirements (as expressed in the Ina Jo criteria). This is accomplished by using the existing Formal Development Methodology (FDM) tool suite and treating the encryption protocol specification like any Ina Jo formal specification.

The two efforts are similar in that they both use a formal notation to express the protocol (The Interrogator uses a Prolog specification.). The use of the Inatest tool for testing particular scenarios is also similar to the use of the Interrogator tool. However, there are at the same time major differences between using Inatest to test a protocol and using the Interrogator. When using Millen's Interrogator the prolog program exhaustively searches for penetrations. Inatest, in contrast, does not search through a large number of scenarios to detect a vulnerability. It is the task of the human analyzer to come up with a possible scenario that is then checked using the Inatest tool to execute the formal specification. Inatest does not direct the analyst to determine what tests to try; it merely aids the analyst by keeping track of state information and performing reductions when possible. Finally, the Interrogator tool was built explicitly for analyzing encryption protocols, but Inatest was built to execute any Ina Jo specification. As a result, the Interrogator includes a more sophisticated display that dynamically illustrates the progress of the protocol being tested.

Conclusions

This paper has proposed an approach to analyzing encryption protocols using existing formal specification and verification techniques. The approach assumes the availability of encryption algorithms that satisfy the properties expressed in the axioms.

An example system was specified using the Ina Jo specification language. Some problems discovered when attempting to prove the original specification are discussed, and a weakness in the formal specification that was revealed by using an interactive testing tool was presented.

One of the advantages of this approach is that the cryptographic facility can be analyzed assuming different encryption algorithms by replacing the set of axioms that express the properties assumed about the encryption algorithms with a new set of axioms that express the properties of a different encryption algorithm.

Another advantage is that the properties of a cryptographic facility can be tested before it is built by using the formal specification and the available interactive testing tool as a rapid prototype.

The flaw that was confirmed by using the Inatest tool was a previously known weakness of the protocol being analyzed. The true worth of the proposed approach will be established only when a flaw can be discovered in a protocol that has been previously assumed to be secure.

References

[Bri 85] Brickell, Ernest F. "Breaking Iterated Knapsacks," *Advances in Cryptology: Proceedings of Crypto 84*, Lecture Notes in Computer Science, Springer-Verlag, New York, 1985.

[BMP 86] Brickell, E.F., J.H. Moore, and M.R. Purtill, "Structure of the S-Boxes of the DES," *Proceedings of CRYPTO 86*, Santa Barbara, California, August 1986.

[CDL 85] Crow, J., D. Denning, P. Ladkin, M. Melliar-Smith, J. Rushby, R. Schwartz, R. Shostak, and F. von Henke, "SRI Verification System Version 2.0 Specification Language Description," SRI International Computer Science Laboratory, Menlo Park, California, November 1985.

[Dav 81] Davies, Donald W., "Some Regular properties of the 'Data Encryption Standard' Algorithm," Proceedings of CRYPTO 81, *Advances in Cryptography*, Department of Electrical and Computer Engineering Report, ECE 82-04, Santa Barbara, California, August 1986.

[EK 85] Eckmann, Steven T., and Richard A. Kemmerer, "INATEST: An Interactive Environment for Testing Formal Specifications," Third Workshop on Formal Verification, Pajaro Dunes, California, February, 1985,
ACM - Software Engineering Notes, Vol. 10, No. 4, August 1985.

[GDS 84] Good, D.I., B.L. DiVito, and M.K. Smith, "Using the Gypsy Methodology," Institute For Computing Science, University Of Texas, June 1984.

[Hol 87] Holzmann, Gerard J., "Automated Protocol Validation in Argos: Assertion Proving and Scatter Searching," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 6, June 1987.

[IEE 82] Sunshine, Carl A. (Editor), Special Issue on Protocol Specification and Verification, *IEEE Transactions on Communications*, Vol. COM-30, No. 12, December 1982.

[Kem 85] Kemmerer, Richard A., "Testing Formal Specifications to Detect Design Errors," *IEEE Transactions on Software Engineering*, Vol. SE-11, No. 1, January 1985.

[LS 84] Lam, Simon S., and A. Udaya Shankar, "Protocol Verification Via Projections," *IEEE Transactions on Software Engineering*, Vol. SE-10, No. 4, July 1984.

[MM 80] Meyer, Carl H., and Stephen M. Matyas, *Cryptography*, John Wiley, 1980.

[MCF 87] Millen, Jonathan K., Sidney C. Clark, and Sheryl B. Freedman, "The Interrogator: Protocol Security Analysis," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 2, February 1987.

[Rob 79] Robinson, L., "The HDM Handbook, Vol I: The Foundations Of HDM," Computer Science Laboratory, SRI International, Menlo Park, California, June 1979.

[RW 83] Rudin, H., and C.H. West (Editors), *Protocol Specification, Testing, and Verification III* Elsevier Science Publishers B.V., North-Holland, 1983.

[SAM 86] Scheid, J., S. Anderson, R. Martin, and S. Holtsberg, "The Ina Jo Specification Language Reference Manual," SDC document, System Development Corporation, Santa Monica, California, January 1986.

[Sim 85] Simmons, G.J., "How to (Selectively) Broadcast a Secret," Proceedings IEEE Symposium on Security and Privacy, Oakland, California, April 1985.

[STE 82] Sunshine, Carl A., David H. Thompson, Roddy W. Erickson, and Susan L. Gerhart, "Specification and Verification of Communication Protocols in AFFIRM Using State Transition Models," *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 5, September 1982.

[TE 81] Thompson, D.H. and R.W. Erickson, eds. "Affirm Reference Manual," USC Information Sciences Institute, Marina del Rey, California, February 1981.

Appendix Formal Specification of the Example System

```
SPECIFICATION Crypto
LEVEL Top_Level

TYPE
   Text,
   Key subtype Text,
   Pos_Integer = T" i:Integer (i>0),
   Information = Set Of Text

CONSTANT
   Num_Terminals: Pos_Integer,
   KMH0, KMH1: Key,
   Encrypt(Key,Text): Text,
   Decrypt(Key,Text): Text

TYPE
   Terminal_Num = T" t:Pos_Integer (t<=Num_Terminals)

CONSTANT
   Terminal_Key(Terminal_Num): Key,
   Terminal_Keys: Information =
      {k:Key ( ∃ t:Terminal_Num (k=Terminal_Key(t)))}

AXIOM
   ∀ t:Text, k1,k2:Key (
      k1=k2 → Decrypt(k1,Encrypt(k2,t))=t )

AXIOM
   ∀ t:Text, k1,k2:Key (
      k1=k2 → Encrypt(k1,Decrypt(k2,t))=t )

VARIABLE
   Session_Key(Terminal_Num): Key,
   Keys_Used: Information,
   Intruder_Info: Information

DEFINE
   Session_Keys: Information ==
      {k:Key ( ∃ t:Terminal_Num (k=Session_Key(t)))}

CRITERION
   ∀ k:Key (k ∈ Intruder_Info → k ∉ Keys_Used)

INITIAL
   Keys_Used=Terminal_Keys ∪ Session_Keys ∪ {KMH0,KMH1}
  & Intruder_Info =
      {ks:Key ( ∃ t:Terminal_Num (ks=Encrypt(KMH0,Session_Key(t))))}
    ∪ {kt:Key ( ∃ t:Terminal_Num (kt=Encrypt(KMH1,Terminal_Key(t))))}
  & ∀ k1,k2:Key (k1 ∈ Intruder_Info & k2 ∈ Keys_Used → k1≠k2)
```

```
Transform ECPH(K:Key, T:Text)  EXTERNAL
 Effect
   N"Intruder_Info =
      if T ∈ Intruder_Info & K ∈ Intruder_Info
         then Intruder_Info ∪ {Encrypt(Decrypt(KMH0,K),T)}
         else Intruder_Info

Transform DCPH(K1:Key, T1:Text)  EXTERNAL
 Effect
   N"Intruder_Info =
     if T1 ∈ Intruder_Info & K1 ∈ Intruder_Info
         then Intruder_Info ∪ {Decrypt(Decrypt(KMH0,K1),T1)}
         else Intruder_Info

Transform RFMK(K1:Key, K2:Key)  EXTERNAL
 Effect
   N"Intruder_Info =
     if K1 ∈ Intruder_Info & K2 ∈ Intruder_Info
        then Intruder_Info ∪ {Encrypt(Decrypt(KMH1,K1), Decrypt(KMH0,K2))}
        else Intruder_Info

Transform Generate_Session_Key(Ter:Terminal_Num)  EXTERNAL
 Effect
   ∃ k:Key ∀ t:Terminal_Num (
       Encrypt(KMH0,k) ∉ Keys_Used
     & Encrypt(Terminal_Key(Ter),k) ∉ Keys_Used
     & k ∉ Keys_Used
     & N"Session_Key(t) =
         if t=Ter
           then k
           else Session_Key(t)
     & N"Keys_Used = Keys_Used ∪ {k}
     & N"Intruder_Info = Intruder_Info ∪
           {Encrypt(KMH0,k),Encrypt(Terminal_Key(Ter),k))}

END Top_Level
END Crypto
```

Cryptosystems based on an analog of heat flow

G. R. Blakley

William Rundell

Department of Mathematics
Texas A&M University
College Station, Texas 77843-3368

CONTENTS

1. Introduction.

It is commonplace to base computer security and information security on hard problems. Recent cryptosystems have been based on the knapsack problem [DE83, pp. 118-126; BR85] and the problem of factoring an integer [DE83, pp. 104-109]. The former problem is NP complete [GA79, p. 247]. The place of the latter problem in complexity theory is not well understood, but it has been around in number theory for a long time.

We will base a family of discrete cryptosystems and continuous scramblers on ideas related to ill-posed problems in differential equations. Once again, complexity theory has little to say about the difficulty of such problems. But they have been around in analysis for a long time.

Our approach will be to proceed from a toy system through a series of more realistic systems, taking up various considerations of decodability, bandwidth expansion, intersymbol dependence, computational expense, and security as they arise. One feature of this approach which will be evident from the outset is its neutrality between the discrete and the continuous realms in communication security. In this respect it is much in the spirit of [BL85, BL86, BL87], which treat various information-theoretic objects in a way designed to minimize dependency on any finiteness properties they may have. More specifically, it is an approach to cryptosystem design closely akin to that recently taken by Davida, Gilbertson and Walker [DA86].

The paper begins by examining partial differential equations (PDEs) suggested by the time evolution of the distribution of temperature in a rod. The plaintext is the initial conditions (i.e. the temperature profile u at $t = 0$), and the cryptext is the $t = 1$ temperature profile. The key is, in a sense to be made clearer below, largely the PDE itself. The role of the boundary conditions is important too. In the end we will not confine ourselves to Dirichlet (fixing a solution u at the end points of the rod) or Neumann (fixing u_x at the end points of the rod) problems. But for a while we will restrict our consideration to the Dirichlet conditions:

$$u(0,t) = u(1,t) = 0 \qquad \text{for all } t \geq 0\,.$$

It is important to note at the outset that security dictates the introduction of nonlinearities at many points. In our particular case this entails the use of nonlinear partial differential operators. Clarity of exposition, on the other hand, calls for casting the discussion herein in linear terms whenever possible. This engages the reader's intuition more readily, and enables us to bring a larger variety of theoretical results to bear on the subject matter. The inevitable compromise this entails should not mislead anyone. Any good implementation of these ideas will be nonlinear through and through, despite any linear biases in the exposition below.

2. Plaintext and encoded text as related time slices through a solution of a PDE.

Throughout this section we largely ignore secrecy considerations, and simply speak of encoding and decoding a message f. By a message we mean a member f of C^D, i.e. a function

$$f : D \to C$$

whose domain D and codomain C have some useful structure, usually group-theoretic. In cryptography and error control, D is often a finite set of symbols. Thus the message

HELLO

is the function

$$f : \{1,2,3,4,5\} \to \{A, B, \ldots, Z\}$$

defined by setting

$$f(1) = \mathrm{H}$$
$$f(2) = \mathrm{E}$$
$$f(3) = \mathrm{L}$$
$$f(4) = \mathrm{L}$$
$$f(5) = \mathrm{O}.$$

An encode/decode pair (c, d) is a pair of functions

$$c : C^D \to E$$
$$d : E \to C^D$$

such that $d(c(f)) = f$, or at least such that $d(c(f))$ is usually fairly near f. E is some appropriate set of encoded messages. Often encoded messages look just like plaintext messages. In this important case, of course, we have

$$E = C^D.$$

Consider an encode map

$$C : R^{[0,1]} \to R^{[0,1]}$$

defined as follows. Take a function $u = u(x,t)$ whose domain is the semi-infinite strip $[0,1] \times [0,\infty)$ and let the plaintext f be the initial condition

$$u(x,0) = f(x) \qquad x \in [0,1] \tag{2.1}$$

for the classical heat equation

$$u_t - u_{xx} = 0 \tag{2.2}$$

subject to the boundary conditions

$$u(0,t) = u(1,t) = 0 \qquad t \geq 0\,. \tag{2.3}$$

Now let u be the unique solution of (2.1), (2.2), (2.3). Then the encoded text g corresponding to the plaintext f is just the restriction of the solution u to the set $\{\,(x,t) \,:\, t = 1\,\}$, i.e.

$$g(x) = u(x,1) \qquad x \in [0,1]\,.$$

Somebody who would rather encode a discrete message than a continuous one can use the same technique after source coding in some appropriate manner. One simple system is to use ASCII code, or some similar code, or to assign integer values to symbols. It is necessary to decide on a block length N, and to partition $[0,1]$ into N subintervals. After that, the plaintext is a function whose value on the jth subinterval is the integer corresponding to the jth symbol in the block. For example, one way to source code is by the correspondence

$$\left.\begin{aligned} \text{blank} &\leftrightarrow 0 \\ \text{A} &\leftrightarrow 1 \\ \text{B} &\leftrightarrow 2 \\ &\vdots \\ \text{Z} &\leftrightarrow 26. \end{aligned}\right\} \tag{2.4}$$

Then it is possible to choose block size $N = 8$. The plaintext message HELLO can be represented by the piecewise continuous function $f \,:\, [0,1] \leftrightarrow \{0,1,2,\ldots,26\}$ such that

$$\begin{aligned} f(x) &= 0, & x &\in [0,1/8) \\ &= 8, & x &\in [1/8,2/8) \\ &= 5, & x &\in [2/8,3/8) \\ &= 12, & x &\in [3/8,5/8) \\ &= 15, & x &\in [5/8,6/8) \\ &= 0, & x &\in [6/8,1] \end{aligned}$$

Here the eight subintervals have been chosen to be of equal length, $1/8$. This is not necessary. Also, $f(x)$ has been modified to satisfy the homogeneous boundary conditions $f(0) = f(1) = 0$.

So, at this point there is a way to represent a (continuous or discrete) plaintext message as a function

$$u(x,0) = f(x)$$

defined on the closed unit interval [0,1] and to produce a corresponding (continuous) encoded message

$$u(x,1) = g(x).$$

3. Intersymbol dependence, smearing, and diffusion in a one-way function based on the heat equation

The encoding of the previous section has the feature that small changes in the plaintext message f cause changes in the encoded message g which are spread throughout g. There is, in the terminology of classical cryptography, "strong intersymbol dependence". A mathematical term for this is "smearing", and it is referred to physically as "diffusion". We will use whatever one of these expressions seems appropriate to the context below. There are provable lower bounds on the extent of this intersymbol dependence, a point to which we shall return in Section 10 below.

Parabolic PDEs such as (2.2) or (2.4) obey the strong maximum principle [RU76]. Thus they enable a cryptosystem designer to produce provable, and sometimes very precise, estimates of diffusion (intersymbol dependence). The physical idea behind this is straightforward, though unrealistic, since energy would have to be transmitted faster than light. In elementary texts [BO77, pp. 511-515] students read about the ordinary heat equation

$$u_t - u_{xx} = 0. \tag{3.1}$$

They discover that it has the feature that, when a torch is applied to a point x_0 on a cold rod at time $t = 0$, every point of the rod is warm [BO77, p. 489] (some warmer than others) at every subsequent time ($t > 0$). The same infinitely fast heat propagation property characterizes generalizations such as

$$u_t - u_{xx} + a(x)u = 0 \tag{3.2}$$

Mathematically, this says that if $f(x) \geq 0$ for $x \in [0,1]$, and is nontrivial then $g(x) > 0$ for $x \in (0,1)$. Suppose that f is zero, except in some small neighborhood $(x_0 - \epsilon,\ x_0 + \epsilon)$ of the point x_0 (Cryptographically this f could correspond to the difference between two plaintext messages which agreed everywhere but at a single symbol position, say the i^{th} symbol in a block). For the heat equation (3.1) the maximum value of $u(x,t)$ for $x \in [0,1]$ will continue to occur at x_0 for all positive t. However the introduction of the function $a = a(x)$ can change the location of this maximum.

Strong intersymbol dependence is not, by itself, sufficient to make an encoding process cryptographically strong. Error control codes also have this feature, but possess little cryptographic strength. And, in fact, the encoding above is not a cryptosystem. It is a fixed process relying on a known equation, the heat equation. There is no secret key material which a sender and a receiver can share before they begin using this encoding as the basis of a secure communications system.

But is the encoding process of this section a one-way function? It would appear to be. There is no known feasible way to turn the process around, so as to proceed from knowledge of an encoded message g to produce the plaintext message f which gave rise to g. The heat equation does not "reverse in time". We will expand on this point toward the end of the next section.

At this point we have the rudiments of a possible alternative to Purdy's [PU 74] high security login scheme.

4. Introducing key dependence into the encoding process. Ill-posed problems.

We will now assume that the sender and receiver get together in secret to modify the heat equation as the basis of an encoding process. Suppose they replace it by the equation

$$u_t - u_{xx} + au = 0, \tag{4.1}$$

where $a = a(x)$ is a function of position only and, as before, $u = u(x,t)$ is a function of position and time. The function $a = a(x)$ is a secret, shared only by the designated encoder (i.e. sender) and the designated decoder (i.e. receiver) who plan to use this encoding process as the basis of a true cryptosystem. Physically, $a\,u = a(x)u(x,t)$ is a reaction term proportional to the value of u. Here we have a position-dependent (but time-independent) proportionality constant a. The function a plays the role of a secret cryptographic key.

Somebody who wants the key material more thoroughly mixed into the encryption process can go beyond (4.1) to

$$\sigma u_t = (cu_x)_x - au \tag{4.2}$$

Here $\sigma = \sigma(x)$ is a function of position only, which corresponds to specific heat. Also $c = c(x)$ is a function of position only, which corresponds to conductivity. Such functions c and a will clearly influence the temperature distributions $u(x,t)$ subsequent to the initial condition

$$u(x,0) = f(x),$$

and cause them to differ from what would be observed in a homogeneous rod with the same initial condition. An obvious change of variable leads to recasting (4.2) in the form

$$u_t = Lu \tag{4.3}$$

where L is the linear operator defined by setting

$$Lu = (pu_x)_x - qu \tag{4.4}$$

Here $p = p(x)$ and $q = q(x)$ are functions of position only, and $p(x)$ is strictly positive for every $x \in [0,1]$.

The introduction of key material moves the original heat-equation-encoding process over toward the cryptographic realm. But it is still deficient in one respect. All the PDEs (2.2), (3.2), (4.2), and (4.4) are parabolic. Consequently the problem of decoding remains ill-posed. In fact we are faced, at this point, with the classical backwards heat flow problem. The backwards parabolic equation is perhaps the canonical example of an ill-posed problem in PDEs [PA74]. J. Hadamard [HA23] initiated the custom of calling a problem *well-posed* if it satisfied the three criteria:

(1) There exists a solution to the problem;
(2) The solution is unique;
(3) The solution depends continuously on the boundary data.

A problem is *ill-posed*, in Hadamard's terminology, if it is not well-posed.

Let us look at the Fourier expansion

$$u(x,t) = \Sigma a_n e^{-n^2\pi^2 t} \sin(n\pi x)$$

of the solution of the problem (2.1) -(2.3), where

$$f(x) \sim \Sigma a_n \sin(n\pi x).$$

Evidently $u(x,t)$ is analytic in x for any $t > 0$. If there is to be any hope of recovering even an $L^2[0,1]$ function from a cryptext

$$g(x) = u(x,1),$$

it must be true a priori that g is analytic (and its Fourier coefficients must decay exponentially fast to zero). Worse still, the sequence

$$u_n(x,t) = (1/n^3)\sin(n\pi x)e^{(1-t)n^2\pi^2}$$

$(n = 1,2,3,...)$ satisfies (2.1) and (2.2), as well as (2.3) with

$$f_n(x) = (1/n^3)e^{n^2\pi^2}\sin(n\pi x).$$

The functions $g_n(x)$ are in this case

$$g_n(x) = (1/n^3)\sin(n\pi x).$$

They tend uniformly to zero in any norm one might conceivably impose on the problem. On the other hand the sequence $\{f_n\}$ cannot converge, in any reasonable norm. In consequence no error, no matter how small, in the codetext can be guaranteed to produce a bounded error in the decoded approximation to the plaintext, even if it were possible to produce such a decoded approximation.

Is there an alternative heat flow model which possesses the main features (certainly including the maximum principle) of equation (4.3) and yet allows time to progress in both directions? The answer is yes, and leads to our next change of the driving equation. This time there is a model which makes decoding, as well as encoding, possible.

5. Decoding an encoded message g to recover an approximation to the plaintext message f.

Consider the pseudoparabolic [SH70a] PDE

$$(L - I)u_t + Lu = 0 \tag{5.1}$$

where I is the identity map and L is an appropriate differential operator. One example might be the linear operator L defined in (4.4). A designated decoder knowing L, the boundary conditions (2.3), and the encoded message

$$g(x) = u(x, 1) \tag{5.2}$$

can hope to recover a good approximation to the plaintext message

$$f(x) = u(x, 0) \tag{5.3}$$

which represents the initial condition (2.1).

Pseudoparabolic PDEs have received much study in the last 15 years. They arise in a variety of problems where it is appropriate to add a certain type of correction term to a parabolic PDE. Examples are problems dealing with second order fluids, with the seepage of fluid through fissured rock, or with certain two-temperature theories of thermodynamics. Perhaps the best reference to the general features of these equations is the paper [SH70a] of Showalter and Ting. See also [SH70b]. On any of the usual function spaces in which one would usually pose (5.1), the operator $L - I$ turns out to be invertible if $q(x) \geq -1$ on $[0, 1]$. Examples of such spaces are the Sobolev space ${H_0}^1 \cap H^2$ or the Schauder space $C_0^{2+\alpha}$. On such spaces the operator $(L - I)^{-1}L$ has an extension to a bounded operator on all of L^2 or of $C^{0+\alpha}$. These considerations lead to the abstract differential equation

$$u_t + (L - I)^{-1}Lu = 0. \tag{5.4}$$

Coupled with the boundary condition (2.3) and the initial condition (2.1), it has a solution

$$u(x, t) = \exp(-t[L - I]^{-1}L)f(x) \tag{5.5}$$

where

$$\{\exp(-t[L - I]^{-1}L) : t \text{ is a real number } \}$$

is a group of operators generated by $[L - I]^{-1}L$ (see [SH70a; SH70b]). If

$$g(x) = u(x, 1)$$

then

$$g = \exp(-(L - I)^{-1}L)f. \tag{5.6}$$

Hence f can be recovered from g by means of the relation

$$f = \exp((L - I)^{-1}L)g$$

The positivity of these two exponential transformations can be shown using the maximum principle for pseudoparabolic partial differential equations [RU76].

6. Computational procedures and costs in the linear case.

Linear and affine maps are abhorrent to cryptosystem designers. Nevertheless it is instructive to examine encryption and decryption calculations in the linear case to show how pseudoparabolic PDEs differ from parabolic, and to get a jumping-off place from which to examine the nonlinear case later. Let L be a linear ordinary differential operator involving only differentiation with respect to the position variable x, for example as in (4.4). Consider the linear pseudoparabolic partial differential equation

$$u_t + (L - I)^{-1}Lu = 0 \tag{6.1}$$

coupled with the initial condition (2.1).

The plaintext message

$$f(x) = u(x, 0) = \exp(0)f(x)$$

gives rise to the cryptext message

$$g(x) = u(x, 1) = \exp(-[L - I]^{-1}L)f(x)$$

and vice versa. Thus

$$\begin{aligned} f(x) &= If(x) \\ &= \exp([L - I]^{-1}L)\exp(-[L - I]^{-1}L)f(x) \\ &= \exp([L - I]^{-1}L)g(x). \end{aligned}$$

The most simple-minded way to structure the computation of g, given f, and the computation of f, given g, is the following. Under the transformation

$$v(x, t) = e^t u(x, t) \tag{6.2}$$

the equation (6.1) becomes

$$v_t + (L - I)^{-1}v = 0 \tag{6.3}$$

Suppose that $y(x)$ satisfies the ordinary differential equation

$$(L - I)y(x) = -h(x) \quad 0 < x < 1 \tag{6.4}$$

$$y(0) = y(1) = 0$$

then $y(x)$ can be written as

$$y = -(L-I)^{-1}h = -\int_0^1 G(x,s)h(s)\,ds \equiv \mathcal{G}h \tag{6.5}$$

where $G(x,s)$ is the Green's function for the equation (6.4). Noting that $v(x,0) = f(x)$ and $g(x) = e^{-t}v(x,1)$ we see that

$$g = e^{-1}\exp(-(L-I)^{-1})f \tag{6.6}$$

and since $\mathcal{G} = -(L-I)^{-1}$ is a bounded operator

$$g = e^{-1}(I + \mathcal{G} + \mathcal{G}^2/2! + \ldots + \mathcal{G}^n/n! + \ldots)f \tag{6.7}$$

Thus g can be computed by truncating the exponential series. The k^{th} term of this series consists of k consecutive integrations of the function $f(s)$ with the Green's function $G(x,s)$. (This last function of course depends on $p(x), q(x)$). It is this integration process that provides the "smearing" of the function $f(x)$.

In (5.1) the operator L need not be the same in both positions, we could equally well have chosen the equation

$$Mu_t + Lu = 0 \tag{6.8}$$

where M is of the same form as L. However if we wish to retain the maximum principle, and for our cryptosystem this is certainly the case, then M and L can only differ in their non-differentiated terms [RU79].

To further expose the action of our method we shall analyze a particularly simple case of (6.8). This example will be used again in later sections to demonstrate further properties. Choose M to be the differential operator $\frac{d^2}{dx^2}$ and L to be $\frac{d^2}{dx^2} + a(x)$, the domain being those $C^2[0,1]$ functions that vanish at $x = 0$ and $x = 1$. The function $u(x,t)$ will thus satisfy

$$u_{xxt} + u_{xx} + a(x)u = 0 \tag{6.9}$$

with

$$u(x,0) = f(x), \quad u(0,t) = u(1,t) = 0 \tag{6.10}$$

Again the change of variable $v = e^{-t}u$ will put the equation in a slightly easier form,

$$v_{xxt} + a(x)v = 0 \tag{6.11}$$

or in the abstract formulation

$$\begin{aligned} v_t + B^{-1}v &= 0 \\ v|_{t=0} &= f(x) \end{aligned} \tag{6.12}$$

where B denotes the operator $Bu = M^{-1}(a(x)u(x))$.

In this case (6.5) takes the form

$$\mathcal{G}h = -\int_0^1 G_0(x,s)a(s)h(s)ds \tag{6.13}$$

where G_0 is the Green's function for M, and is given by

$$G_0(x,s) = \begin{cases} (1-s)x & x \le s \\ (1-x)s & x \ge s \end{cases}$$

In each successive application of $\mathcal{G}$ that is required to approximate $g(x)$ by (6.7), we see that the coefficient $a(x)$ (our chosen key) comes in to modify the input function by integration against $a(x)$.

The above analysis was presented only to show the workings of the encryption operator and its dependence on the coefficient $a(x)$. In practice one would not use a power series method, but rather a finite difference scheme based, for example, on the Crank-Nicholson method [YO73, pp. 1078, 1086-1088].

It must be emphasized that the decryption process is entirely the same as encryption

$$\begin{aligned} \text{codetext} &= \exp(-A)\ (\text{plaintext}) \\ \text{plaintext} &= \exp(A)\ (\text{codetext}) \end{aligned}$$

since both initial and final value problems are solved in the same manner for the equation (5.1).

7. Cryptosystems based on nonlinearities in pseudoparabolic PDEs.

If $p = 1$ in L as in (4.4), and the transformation $u \to ue^{-t}$ is performed on (5.1) then, as in section 6, we arrive at the simplest form of our equation

$$u_{xxt} - u_t + q(x)u = 0 \tag{7.1}$$

Even in this basic case it is not easy to find $q(x)$ from a knowledge of both $f(x)$ and $g(x)$. Problems of this kind are called undetermined coefficient problems and in most cases are notoriously ill-posed. There may be an infinite number of $q(x)$ that would yield the same $g(x)$ from a given $f(x)$. Even if this were not so, there may be two functions $q_1(x)$ and $q_2(x)$ that take a given plaintext onto very similar codetexts, yet for another plaintext, the corresponding codetexts would not be close. If $q = q(x,t)$ were allowed to be time-dependent then it would, at least in theory, be impossible to obtain this function of two variables by giving only one additional function g at the single variable x.

Equations of the form (6.6) are simply matrix equations of size N by N. Such equations could be solved for the eigenvectors of the matrix given a sufficient quantity of plaintext-codetext pairs. Even if it were not possible to recover the key, it might be possible to read the messages.

This leads to the final modification in our cryptosystem, the addition of some nonlinear terms. Equations of the type (4.3) or (5.1) are referred to as "diffusion equations". This terminology is based on one of their features. They spread initially localized heat throughout the body — the very property that gives us our intersymbol dependence. The addition of a reaction term F, if correctly chosen, can tend to counteract this diffusing tendency by further increasing the temperature at places where $u(x,t)$ is already large. These combination equations are referred to as "reaction-diffusion" equations. Our chosen type of equation reads

$$(L - I)u_t + Lu = F, \tag{7.2}$$

where the function F may depend on x, t, u and u_x. In (7.2) one could also assume that the coefficients of L depend on x, t, u, u_x and the maximum principle would still hold (under suitable restrictions on sign) as would the invertibility of (7.2) in time. The boundary conditions $u(0,t) = u(1,t) = 0$ could be generalized to conditions of the form $u_x + h(t)u = \beta(t)$ where h and β could form part of the key. They could also be made nonlinear.

8. Computational considerations regarding encryption and decryption in the nonlinear case.

How would one solve (7.2), how would the key enter, and what is the additional expense in computation?

Let us first consider a possible attack on the problem that uses the discussion in section 6 as a basis. If $v(x,t)$ satisfies

$$\begin{aligned} (L-I)v_t + Lv &= 0 \\ v(x,0) &= f(x) \end{aligned} \tag{8.1}$$

subject to the usual boundary conditions (2.3) at $x=0$ and $x=1$, then we can write the solution in abstract form

$$v = \exp(tA)f$$

with $A=(L-I)^{-1}L$. Of course if we know the coefficients of L we can construct the operator $\exp(-tA)$.

If $u(x,t)$ now satisfies (7.2) then

$$\begin{aligned} u_t + Au &= (L-I)^{-1}F[u] := \hat{F}(u) \\ u(x,0) &= f(x) \end{aligned} \tag{8.2}$$

where $\hat{F}$ depends on (x,t,u,u_x) and the solution to this can be written in the form

$$u(t) = \exp(-tA)f + \int_0^t \exp(-(t-\tau)A)\hat{F}(u(\tau))d\tau \tag{8.3}$$

We can consider (8.3) as a nonlinear integral equation for u whose free term is $\exp(-tA)f$ and whose kernel is $K(t,\tau,u) = \exp(-(t-\tau)A)\hat{F}$. The equation can be solved by the method of successive approximations under mild conditions on the function F (smoothness in x,t, Lipschitz continuity in the variables u,u_x). The approximation scheme starts with an initial guess $u_0(t)$ (usually one uses the free term; $\exp(-tA)f$ in this case) and then updates by

$$u_{n+1}(t) = \exp(-tA)f + \int_0^t \exp(-(t-\tau)A)\hat{F}(u_n(\tau))d\tau \tag{8.4}$$

for $n \geq 0$.

In practice one would not invoke the machinery in quite this form. A finite difference scheme would again be used and the nonlinear term would be evaluated by successive approximations in a subloop. This subloop is usually quite short — typically four or five iterations suffice. For a given mesh size, the cost of adding nonlinearities is roughly a fixed amount (independent of the mesh size) times the cost of the linear case.

9. Resistance of the nonlinear pseudoparabolic PDE cryptosystem to chosen plaintext attack.

Even for the simple model problem (6.11) it is not known whether one can recover the key material $a(x)$ from a knowledge of $f(x)$ and $g(x)$. If, of course, enough message pairs are intercepted then eventually, since the problem is linear, it is possible for a cryptanalyst to find the action of the system on each element of a basis for the set of possible plaintexts. This could then be used to read subsequent messages. In the case of a nonlinear version of the system this method is no longer applicable. The possibility of recovering a function F of the form $F(x,t,u,u_x)$ from measurements of f,g pairs lies outside the scope of present research in the area of undetermined coefficient problems in partial differential equations at present. And the image of a vector space of plaintext messages will be, at best, a complicated manifold of cryptext messages. So even reading subsequent messages in the absence of key information appears to be a difficult problem.

10. Block size, its effect on security, the cost of encryption and error control

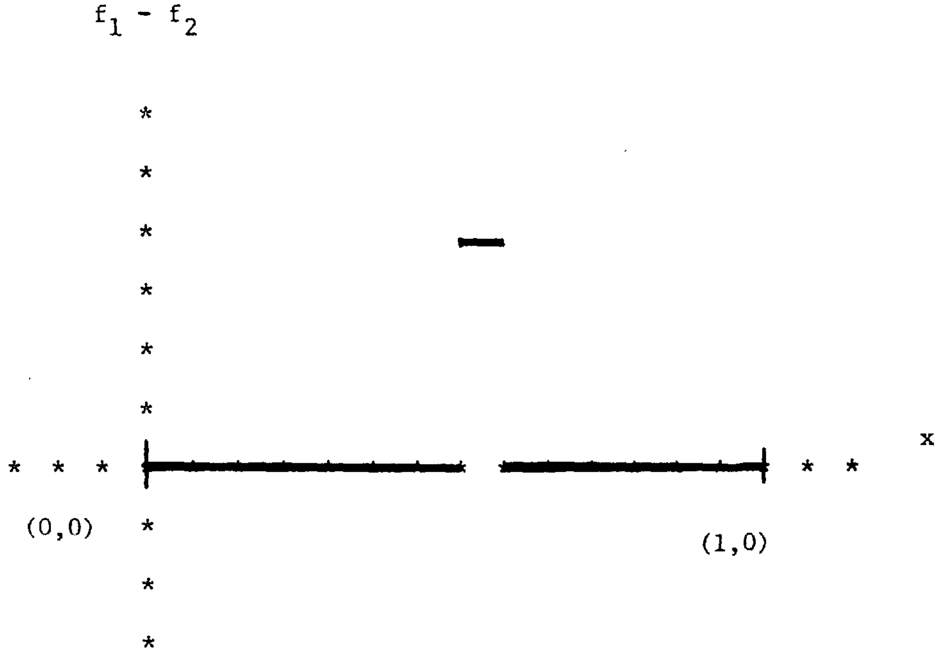
In the discrete case, and for linear models, we can give an indication of what should be the expected dependence of the performance of the system on blocksize.

Suppose two messages f_1 and f_2 differ only at one position. The if we let g_1 and g_2 be the associated codetexts, we see that

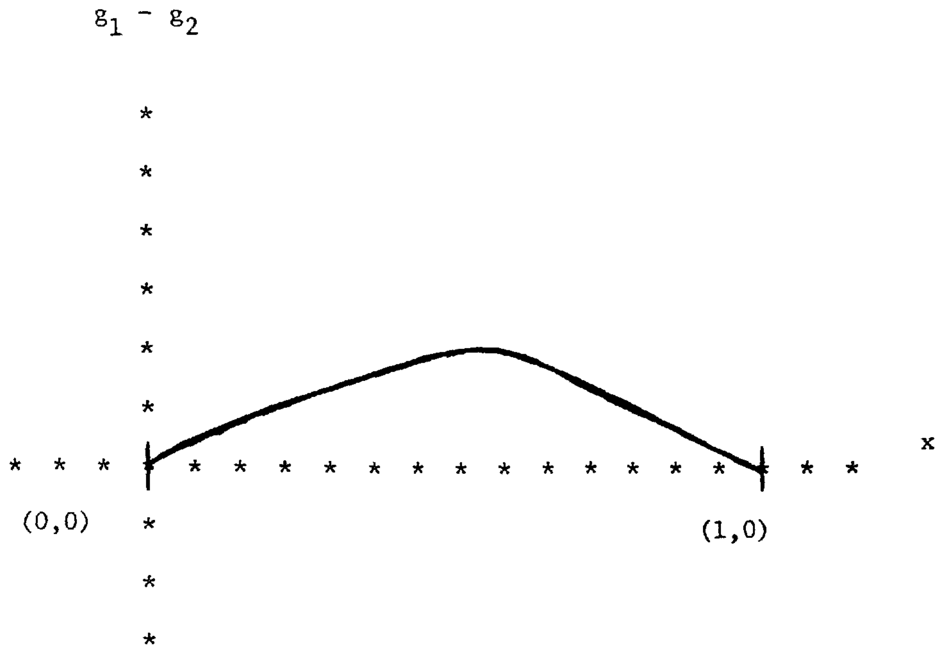
$$g_1 - g_2 = \exp(-tA)(f_1 - f_2)$$

where $f_1 - f_2$ is zero except for the interval, say I, where the difference occurs. See Figure 10.1.

From the physical interpretation of the system as a model of heat flow, one would expect that $g_1 - g_2 \neq 0$ for all x, $0 < x < 1$, and this is in fact guaranteed by the maximum principle for pseudoparabolic equations [RU76]. Thus somebody who kept the codetext g to arbitrary precision would be assured that every change



$f_1 - f_2$
x
(0,0)
(1,0)



$g_1 - g_2$
x
(0,0)
(1,0)

in the plaintext would give rise to a change in every element of the codetext. In practice it is necessary to truncate $g(x)$. To how many places should $g(x)$ be kept? The answer depends on what you expect from the system. It is necessary to retain $g(x)$ to sufficient accuracy so you can invert to recover $f(x)$. For example if $f(x)$ is formed by the scheme (2.4) then an error in $f(x)$ of at most 0.5 is tolerable, and a further margin of safety would be desirable. Standard estimates for differential operators make it possible to calculate the maximum error in $g(x)$ for a given error in $f(x)$, and conversely. The actual factor will, of course, depend fairly strongly on the choice of the coefficients. For example if $a \leq 0$ in (6.11), the error in $f(x)$ obtained by inverting the equation in time will be less than that in $g(x)$. This is a consequence of the maximum principle. For this example, one could thus recover $f(x)$ despite an error in $g(x)$ of as much as ± 0.5.

An important consideration is to retain sufficient accuracy in $g(x)$ so that all (or at least a high fraction) of the values of g at the grid points are still seen to be positive after truncation, when g is the codetext coming from the function $f_1 - f_2$. Can this be estimated? This is in theory possible even in the full nonlinear case, but a practical bound may be exceedingly difficult to obtain. In the case of our simple model (6.11) we can make a reasonable quantifiable attempt. However, it should be stressed that in all cases the value of a $g(x)$ that is produced from such an $f_1 - f_2$ will decrease (linearly) with blocksize. Suppose the change in f represents one of unit height in a blocksize N. Then, if transformed to blocksize nN for $n > 1$, this would represent n changes, each of unit height.

If $a(x) \leq 0$ then from (6.7), (6.13) we see that

$$g(x) = e^{-1}(I + \mathcal{G} + \mathcal{G}^2/2 + \ldots)f$$

and

$$\mathcal{G}h = -\int_0^1 G_0(x,s)a(s)h(s)ds$$

Note that if $h \geq 0$ then $\mathcal{G}h \geq 0$ (a simple consequence of the maximum principle for ordinary differential equations). Thus $\mathcal{G}$ is a positive operator and it follows that

$$g(x) \geq e^{-1}\{f(x) - \int_0^1 G_0(x,s)a(s)f(s)\,ds\} \tag{10.1}$$

If

$$f(x) = \begin{cases} 1 & x_0 \leq x < x_0 + 1/N \\ 0 & \text{otherwise} \end{cases}$$

then, for example, if $x > x_0 + 1/N$

$$\begin{aligned} g(x) &\geq e^{-1} \int_{x_0}^{x_0+1/N} G_0(x,s)a(s)\,ds \\ &= \frac{1-x}{e} \int_{x_0}^{x_0+1/N} s\,a(s)\,ds \end{aligned} \tag{10.2}$$

The above can be used to obtain a lower bound for $g(x)$ at each of the gridpoints $x_i = 1/N, 2/N, \ldots, (N-1)/N$. To guarantee that such a locally supported $f(x)$ gives, in the truncated form of $g(x)$, a nonzero contribution at each of the gridpoints, it is enough to choose the accuracy to exceed the lower bound.

Note that in (10.2) of $a(x) \leq -1$ for all x in [0,1] then

$$g(x) \geq \frac{1-x}{e}\left(\frac{x_0}{N} + \frac{1}{2N^2}\right)$$

so that in the worst case of $x_0 = 1/N$, $x = (N-1)/N$ (looking at the change in g at the right hand boundary from a change in $f(x)$ concentrated at the left hand boundary)

$$g\left(\frac{N-1}{N}\right) \geq \frac{3}{2e}\frac{1}{N^3}$$

this would predict that with $N = 100$ one would require 7 figure accuracy in $g(x)$.

The choice of blocksize will have an effect on computational speed. In section 6 we showed that in the simplest case the calculation of $g = \exp(-A)f$ was equivalent to multiplying the N vector f by an $N \times N$ matrix, the matrix $\exp(-A)$, where N is the blocksize. In the nonlinear case the computational time using the common finite difference schemes is also proportional to N^2. Of course for a given size text, increasing the blocksize by a factor of N puts the text through the encryption process N times as fast, so that there is a linear increase in total computation time with increase in blocksize.

11. Bandwidth expansion in the linear case.

A limited bandwidth function is a function that can be represented as a finite sum of trigonometric polynomials,

$$f(x) = \sum_{n=1}^{M} a_n \sin(n\pi x) \tag{11.1}$$

We have restricted ourselves to sines above, because we have chosen to speak primarily of Dirichlet boundary conditions (2.3). Obviously, more general trigonometric polynomials are possible, as we have noted elsewhere. The reader can easily fill in the details. We shall say a function $f(x)$ has *L-limited bandwidth* if

$$f(x) = \sum_{n=1}^{M} a_n \phi_n(x). \tag{11.2}$$

Here $\{\phi_n(x)\}_{n=1}^{\infty}$ are the normalized eigenfunctions for the operator L defined in (4.4). It acts on $C^2[0,1]$ functions that vanish at $x = 0$, $x = 1$. Note that $\{\sin n\pi x\}_{n=1}^{\infty}$ are the eigenfunctions for the operator $Lu = u''$, with $u(0) = u(1) = 0$. Sturm-Liouville theory guarantees that the eigenvalues $\{\lambda_n\}_{n=1}^{\infty}$ of L obey the asymptotic formula

$$\lambda_n \approx n^2\pi^2 \tag{11.3}$$

and that the eigenfunctions $\phi_k(x)$ have exactly $k-1$ zeroes in $0 < x < 1$.

We claim that the encryption process using (5.1) - (5.3) does not increase the L-limited bandwidth of a function. Let

$$f(x) = \sum_{1}^{M} a_n \phi_n(x)$$

where $\phi_n(x)$ satisfies

$$\left.\begin{aligned} -L\phi_n &= -(p\phi_n')' + q\phi_n = \lambda_n \phi_n \qquad 0 < x < 1 \\ \phi_n(0) &= \phi_n(1) = 0. \end{aligned}\right\} \tag{11.4}$$

Then a simple separation of variables argument shows that if $u(x,t)$ satisfies

$$\begin{aligned} (L - I)u_t + Lu &= 0 \\ u(x,0) &= f(x) \end{aligned}$$

then

$$u(x,t) = \sum_{n=1}^{M} a_n e^{-\lambda_n t/(1+\lambda_n)} \phi_n(x)$$

and hence the codetext $u(x,t) = g(x)$ must equal

$$g(x) = \sum_{n=1}^{M} b_n \phi_n(x), \qquad b_n = a_n e^{-\lambda_n/(1+\lambda_n)}. \tag{11.5}$$

Not only does $g(x)$ have the same L-limited bandwidth as $f(x)$ but, for large n, the energy in each band is of the same order of magnitude

$$\frac{b_n}{a_n} = e^{-\lambda_n/(1+\lambda_n)} \approx e^{-1}e^{1/(1+n^2\pi^2)} \tag{11.6}$$

A transmitter can be thought of as the vibration of a *homogeneous* material, the governing equation of motion being the wave equation, which in one space dimension is

$$u_{tt} - u_{xx} = 0.$$

If we could build an *inhomogeneous* transmitter whose motion was governed by the hyperbolic equation

$$u_{tt} - Lu = 0 \tag{11.7}$$

defined on the domain $0 \le x \le 1$, $0 \le t \le 1$, and with the operator L as before, then the output, instead of being the sum of the eigenfunctions of the operator $\frac{d^2}{dx^2}$ with the corresponding frequencies corresponding to the eigenvalues. $\{n^2\pi^2\}_{n=1}^{\infty}$ will be the eigenfunctions of the operator Lu with the frequencies determined by the eigenvalues $\{\lambda_n\}_{n=1}^{\infty}$.

If a transmitter/receiver pair were to be built using (11.7) as the governing equation then this process would share many of the ideas of our cryptosystem for the linear case. Even if an eavesdropper could determine all the frequencies of vibration of this linear system, (tantamount to knowing all the eigenvalues of L), then this is insufficient to recover the operator L [HO73]. This is a statement of the classical inverse Sturm-Liouville problem. If additional information is given, for example the *energy* in each eigenmode, then recovery methods are possible in one space dimension. In higher space dimensions the determination of L from such spectral data remains an enigma. With current technology it might be possible to reconfigure a transmitter/receiver pair electronically without actually modifying the hardware.

Finally, it should be noted that the bandwidth expansion problem in the non-linear case is difficult to treat theoretically due to the loss of the superposition principle. The maximum principle guarantees that there will be bandwidth expansion, and numerical simulations could be performed in order to obtain quantitative estimates.

12. Numerical examples.

We ran a numerical simulation of the encryption system in a simple case; taking the linear model and restricting our attention to the equation (5.1) with L the operator $\frac{d^2}{dx^2} = q(x)$. We converted alphanumeric plaintext into a piecewise constant function $f(x)$ by means of (2.4), and adjusted the norm of the key $q(x)$ so that the codetext $g(x)$, when evaluated at the gridpoints $x_i = i/N$, lay in the range $0-999$ after rounding off to the nearest integer. This retention in accuracy in $g(x)$ was sufficient to recover the plaintext $f(x)$ in all cases that we ran. The maximum blocksize attempted was $N = 512$. With N in the range $50-100$ we found that a single change in a character in the plaintext changed on average all of the values of the codetext, although some by only one or two numbers. The greatest change was usually near the gridpoint where the change in the plaintext occured, but this was somewhat key-dependent.

When we attempted to decode a message with a key that differed at only one gridpoint from the one used to encrypt, we found that the resulting "plaintext" had changed by one or two numbers in about $1/3$ to $1/2$ of the positions, again concentrated near the position of key change. Blocksize was again in the range $50-100$.

There is no reason why such an approach need be restricted to a single space variable x. In fact fax, photos and other multidimensional messages might more naturally be considered by means of pseudoparabolic PDEs in $\Omega \times [0,1]$ where Ω is some appropriate region in $\mathbb{R}^n$.

13. The need for error control in the discrete case.

It is clear that finite computational resources produce a decryption which is merely close to, not exactly equal to, the plaintext which was originally encrypted. The difference can be enough to change a symbol here and there. This will put a slightly perturbed version of the original plaintext into the receiver's hands. If large block size (of the order of hundreds of bits) is used there is not much overhead expense in applying an agreed-upon error-control coding process to the plaintext before encrypting. If the language in which the plaintext message is written has a fair amount of redundancy this may not be necessary. If that language has almost no redundancy, then very cheap simple cryptosystems are probably adequate to conceal message traffic in it.

14. Discussion

The purpose of this paper has been to demonstrate the feasibility of basing a family of conventional (as opposed to public key) cryptosystems on a circle of hard problems arising in the theory of partial differential equations. We have shown why it is hard to avoid the use of nonlinear pseudoparabolic PDEs and, possibly, of nonlinear boundary conditions in formulating such a cryptosystem. Our methodology is neutral as regards continuous or discrete messages. It seems quite amenable to analog calculations now that there are natural purely analog methods [PE 86] for time reversal of an optical signal.

As often happens in cryptographic discussions, we have actually said only that somebody who knows how to solve interesting and long-standing hard problems (in analysis, in the case of this family of cryptosystems) can break cryptosystems expeditiously. But like other cryptosystem designers we allow ourselves to think that, so far, it looks as if the converse is also true.

This research was supported, in part, by NSA Grant MDA 904-83-H-0002.

15. References

BL85 G. R. Blakley, Information theory without the finiteness assumption, I: Cryptosystems as group-theoretic objects. in *G. R. Blakley and D. Chaum (editors), Advances in Cryptology, Proceedings of Crypto '84*, Springer-Verlag, Berlin (1985), 314-338.

BL86 G. R. Blakley, Information theory without the finiteness assumption, II: Unfolding the DES. in *H. Williams (editor), Advances in Cryptology, Proceedings of Crypto '85*, Springer-Verlag, Berlin (1986), to appear.

BL87 G. R. Blakley and C Meadows, Information theory without the finiteness assumption, III: Data compression and codes whose rates exceed unity. *Proceedings of the 1986 Cirencester Conference on Cryptography and Coding, IMA*, (1987) to appear.

BR85 E. Brickell, Breaking iterated knapsacks, in *G. R. Blakley and D. Chaum (editors), Advances in Cryptology, Proceedings of Crypto '84*, Springer-Verlag, Berlin (1985), 342-358.

DA86 G. I. Davida, C. Gilbertson and G. Walter, Analog cryptosystems, in *Proceedings of Eurocrypt '85*, Springer-Verlag, Berlin (1986), to appear,
also
Technical Report TRCS-84-1. *Department of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee*, (1984).

DE83 D. E. R. Denning, Cryptography and Data Security, Addison-Wesley, Reading, Massachusetts (1983).

FR64 Partial Differential Equations of Parabolic type, Prentice Hall, Englewood Cliffs, New Jersey, (1964).

GA79 M. R. Garey and D. S. Johnson, Computers and Intractibility: A Guide to the Theory of NP-Completeness, Freeman, San Francisco (1979).

HA23 J. Hadamard, Lectures on the Cauchy Problem in Linear Partial Differential Equations, Yale Univ. Press. New Haven, 1923.

HO73 H. Hochstadt, The inverse Sturm-Liouville problem, *Comm. Pure Applied Math.*, **16**, 715–729, (1973).

PA74 L. E. Payne, Improperly Posed Problems in Partial Differential Equations, Springer Lecture Notes, (1974).

PE 86 D. M. Pepper, Applications of optical phase conjugation, *Scientific American*, **254**, No 1, 74–83, (1986).

PR67 M. H. Protter and H. Weinberger, Maximum Principles in Differential Equations, Prentice Hall, Englewood Cliffs, New Jersey (1967).

PU 74 G. B. Purdy, A high security log-in procedure, *Comm. ACM*, **17** 442–445 (1974).

RU76 W. Rundell and M. S. Stecher, Maximum principles for pseudoparabolic partial differential equations, *SIAM J. Math. Analysis*, **7** (1976), 898–912.

RU79 W. Rundell and M. S. Stecher, The nonpositivity of solutions to pseudoparabolic equations, *Proc. Amer. Math. Soc.*, **75**, No. 2 (1979), 251–254.

SH70 R. E. Showalter and T. W. Ting, Pseudoparabolic partial differential equations, *SIAM J. Math. Analysis*, **1** (1970), 1–25.

SH70b R. E. Showalter, Well-posed problems for a partial differential equation of order $2m+1$. *SIAM J. Math. Anal.* **1** (1970) 214–231.

WE65 H. Weinberger, A First Course in Partial Differential Equations, Xerox College Publishing, Lexington, Massachusetts, (1965).

YO73 D. M. Young and R. T. Gregory, A Survey of Numerical Mathematics, Volume 2, Addison-Wesley, Reading, Massachusetts (1973).

A Combinatorial Approach to Threshold Schemes

D. R. Stinson and S. A. Vanstone
University of Manitoba and University of Waterloo

Abstract We investigate the combinatorial properties of threshold schemes. Informally, a (t, w)-threshold scheme is a way of distributing partial information (shadows) to w participants, so that any t of them can easily calculate a key, but no subset of fewer than t participants can determine the key. Our interest is in *perfect* threshold schemes: no subset of fewer than t participants can determine any partial information regarding the key. We give a combinatorial characterization of a certain type of perfect threshold scheme. We also investigate the maximum number of keys which a perfect (t, w)-threshold scheme can incorporate, as a function of t, w, and the total number of possible shadows, v. This maximum can be attained when there is a Steiner system S(t, w, v) which can be partitioned into Steiner systems S(t - 1, w, v). Using known constructions for such Steiner systems, we present two new classes of perfect threshold schemes, and discuss their implementation.

Introduction

Let X be a set of v elements (which we refer to as *shadows*), and let K be a set of m elements (called *keys*). A *(t, w)-threshold scheme* is a pair $(\mathcal{B}, \phi)$, where $\mathcal{B}$ is a set of b (distinct) w-subsets of X (*blocks*), and ϕ: $\mathcal{B} \rightarrow K$, such that the following properties are satisfied:

1) any t shadows determine at most one key (i.e. for every t-subset S of X, $|\{\phi(B): S \subseteq B \in \mathcal{B}\}| = 0$ or 1).
2) any set of fewer than t shadows which occur in a block do *not* determine a unique key (i.e. for every t'-subset S of X where t' < t, $|\{\phi(B): S \subseteq B \in \mathcal{B}\}| \neq 1$).

The idea behind threshold schemes is that we wish to give partial information (shadows) to w people, so that any t of them can determine the key, but no group of fewer than t can do this. Suppose we want to send key K. We then pick a block B such that $\phi(B) = K$, and then give each of the w participants a different shadow in B.

Threshold schemes were first described by Shamir [7] and Blakely [3] in 1979. Since then, many constructions have been given for threshold schemes. Most of the constructions are linear algebraic in nature (see, for example, Kothari [6]). Recently, Beutelspacher [2] has given some constructions for threshold schemes using finite geometries. The purpose of this paper is to investigate the properties of threshold schemes from a combinatorial viewpoint. This more general approach enables us to give some new constructions for threshold schemes based on combinatorial designs.

Property 2) in the definition of threshold schemes says that t participants are required in order to determine the key K, but it is possible that a group of t' (< t) participants may be able to obtain some partial information, if they can rule out certain keys, for example. Ideally, we would like to have threshold schemes where no partial information would be conveyed in this instance.

To make these ideas precise, we discuss the idea of security for threshold schemes (see, for example, Blakely and Meadows [4]). First, we introduce some probability distributions. We assume that we are given a specified probability distribution on the key space K. For every key K, we then choose a probability distribution on the blocks in $\phi^{-1}(K)$. Together, these determine a probability distribution on $\mathcal{B}$.

Now, suppose a block B has been chosen, and the shadows distributed to the participants. Any subset of shadows $S \subseteq B$ defines a conditional probability distribution on K:

$$p(K \mid S) = p(S \mid K) \cdot p(K) / p(S)$$
$$= \sum_{\{B \in \phi^{-1}(K):\ S \subseteq B\}} p(B \mid K) \cdot p(K) / \sum_{K' \in K} \sum_{\{B \in \phi^{-1}(K'):\ S \subseteq B\}} p(B \mid K') \cdot p(K')$$

We now can rigourously define the concept of security in a threshold scheme. Given a (t, w)-threshold scheme, and given t' < t, we say that the threshold scheme is *perfectly t'-secure* if for every subset $S \subseteq X$ of cardinality t' which occurs as a subset of at least one block, and for every key K,we have that $p(K \mid S) = p(K)$.

We will refer to a threshold scheme as *regular* if b / m blocks correspond to each possible key; and for every key K, each block in $\phi^{-1}(K)$ is chosen with equal probability m / b. It follows that, in a regular threshold scheme, we have

$$p(K \mid S) = \sum_{\{B \in \phi^{-1}(K):\ S \subseteq B\}} p(K) / \sum_{K' \in K} \sum_{\{B \in \phi^{-1}(K'):\ S \subseteq B\}} p(K').$$

Given $S \subseteq X$ and K, define $\lambda(S, K) = |\{B \in \phi^{-1}(K):\ S \subseteq B\}|$. It then follows that

$$p(K \mid S) = p(K)\cdot\lambda(S, K) \,/ \sum_{K' \in \mathbf{K}} p(K')\cdot\lambda(S, K').$$

In a regular scheme, $p(K \mid S) = p(K)$ if and only if

$$\lambda(S, K) = \sum_{K' \in \mathbf{K}} p(K')\cdot\lambda(S, K').$$

Thus, a regular threshold scheme is perfectly t'-secure if and only if, for all $S \subseteq X$ of cardinality t', we have that $\lambda(S, K)$ is independent of the key K. Equivalently, we have the following.

Lemma 1.1 A regular (t, w)-threshold scheme $(\mathcal{B}, \phi)$ is perfectly t'-secure if and only if the following property holds: for every $S \subseteq X$ of cardinality t', there exists a non-negative integer $\lambda(S)$, such that, for every key K, we have

$$|\{B \in \phi^{-1}(K):\ S \subseteq B\}| = \lambda(S).$$

The following result is now an immediate consequence.

Lemma 1.2 If a regular threshold scheme is perfectly t'-secure, then it is perfectly t"-secure for all t", $1 \le t'' \le t'$.

Proof: Given a subset T, where $|T| = t''$, we have

$$\lambda(T) = \frac{\sum_{\{S:\ |S| = t' \text{ and } T \subseteq S\}} \lambda(S)}{\binom{w - t''}{t' - t''}}$$

2. A combinatorial characterization of perfect threshold schemes

Our main interest is in regular (t, w)-threshold schemes that are perfectly (t - 1)-secure. We refer to such a threshold scheme as *perfect*. Next, we give a characterization of perfect threshold schemes in terms of the blocks corresponding to each key.

Let $\mathcal{A}$ be a set of w-subsets (*blocks*) of a v-set X. We refer to $(X, \mathcal{A})$ as a ***w-uniform hypergraph***, and we say that v is the number of *points* in the hypergraph. Given any integer $t' \leq w$, define a multiset $\mathcal{A}(t') = \bigcup_{A \in \mathcal{A}} \{S: |S| = t', S \subseteq A\}$. Note that $\mathcal{A}(t')$ can contain "repeated" t'-subsets. We say that $\mathcal{A}(t')$ is the multiset of *induced* t'-subsets of $\mathcal{A}$.

Two w-uniform hypergraphs $(X, \mathcal{A}_1)$ and $(X, \mathcal{A}_2)$ are defined to be *t-compatible* if the following two properties are satisfied:

1) $\mathcal{A}_1(t - 1) = \mathcal{A}_2(t - 1)$, and
2) $\mathcal{A}_1(t) \cap \mathcal{A}_2(t) = \varnothing$.

The following result characterizes perfect (t, w)-threshold schemes in terms of t-compatible w-uniform hypergraphs.

Theorem 2.1 There exists a perfect (t, w)-threshold scheme having v shadows and m keys if and only if there exist m mutually t-compatible w-uniform hypergraphs on v points.

One way to approach the construction of a perfect (t, w)-threshold scheme having v shadows is to start with a multiset $\mathcal{S}$ of (t - 1)-subsets of a v-set, and attempt to find t-compatible w-uniform hypergraphs $\mathcal{A}_1, \ldots, \mathcal{A}_m$ such that $\mathcal{A}_i(t - 1) = \mathcal{S}$, $1 \leq i \leq m$ (that is, so that $\mathcal{S}$ is the multiset of induced (t - 1)-subsets of each $\mathcal{A}_i$). Given t, w, v, and $\mathcal{S}$, we would want to find the maximum number of such hypergraphs (= the maximum number of keys in the resulting threshold system). We denote this number by $m(t, w, v, \mathcal{S})$. Also, denote

$$m(t, w, v) = \max\{m(t, w, v, \mathcal{S}): \mathcal{S} \text{ is a multiset of } (t - 1)\text{-subsets of a v-set}\}.$$

As one would suspect, determining the numbers $m(t, w, v, \mathcal{S})$ are very difficult. Holyer [5] has proved that the question "can a graph G be edge-decomposed into triangles?" is NP-complete. This question can be rephrased as "is $m(3, 3, v, G) \geq 1$?", where G has v vertices. Hence, determining the numbers $m(t, w, v, \mathcal{S})$ is NP-hard.

We can, however, give some upper bounds on $m(t, w, v)$ and $m(t, w, v, \mathcal{S})$, as follows.

Theorem 2.2 Let λ be the largest multiplicity of any block in $\mathcal{S}$. Let u denote the smallest positive integer such that

$$\binom{u}{w-t+1} \geq \lambda.$$

Then $m(t, w, v, \mathcal{S}) \leq (v - t + 1) / u$.

Proof: Let $\mathcal{A}$ be a w-uniform hypergraph $\mathcal{A}$ on v points such that $\mathcal{A}(t-1) = \mathcal{S}$. Let $S \in \mathcal{S}$ have multiplicity λ. The λ blocks in $\mathcal{A}$ which contain S must all be distinct. Hence, together they contain at least u different elements.

Now, suppose we have m t-compatible hypergraphs. From each of the m hypergraphs, we obtain a set of u elements as described above. These m sets must be disjoint, since otherwise the respective multisets of induced t-subsets would not be disjoint. Also, these m sets are disjoint from S. Hence, $m \leq (v - t + 1) / u$.

Corollary 2.3 $m(t, w, v) \leq (v - t + 1) / (w - t + 1)$.

Proof: In order to maximize the bound on m, we minimize u. This occurs when $\lambda = 1$. Then $u = w - t + 1$, and $m \leq (v - t + 1) / (w - t + 1)$.

We can give a nice characterization of when equality can be met in the above bound.

Theorem 2.4 $m(t, w, v) = (v - t + 1) / (w - t + 1)$ if and only if there exists a Steiner system $S(t, w, v)$ which can be partitioned into Steiner systems $S(t - 1, w, v)$.

Proof: First, suppose that we have an $S(t, w, v)$ which can be partitioned into $S(t - 1, w, v)$. The number of these $S(t - 1, w, v)$ is $(v - t + 1) / (w - t + 1)$, and they are t-compatible, so $m(t, w, v) = (v - t + 1) / (w - t + 1)$.

Conversely, suppose $m(t, w, v) = m = (v - t + 1) / (w - t + 1)$. Let $\mathcal{S}$ denote the multiset of induced $(t - 1)$-subsets of t-compatible w-uniform hypergraphs $\mathcal{A}_1, \ldots, \mathcal{A}_m$. Let S be any $(t - 1)$-subset in $\mathcal{S}$. Then, S has multiplicity 1. Also, since $m = (v - t + 1) / (w - t + 1)$, we see that every t-subset of the form $S' = S \cup \{x\}$ occurs exactly once in $\bigcup_{1 \leq i \leq m} \mathcal{A}_i(t)$.

We now prove that *any* t-subset of points S" occurs as an induced t-subset in $\bigcup_{1 \leq i \leq m} \mathcal{A}_i(t)$. Fix some t-subset $S' = S \cup \{x\}$. We prove that S" occurs as an induced t-subset by reverse induction on $|S' \cap S''|$. As an induction assumption, suppose that S" occurs as an induced t-subset exactly once if $|S' \cap S''| \geq t - j$. Clearly, we can start the induction at $j = 0$.

Now, suppose that $|S' \cap S''| = t - j - 1$. Let $y \in S' \setminus S''$ and let $z \in S'' \setminus S'$. Define $S^* = S'' \cup \{y\} \setminus \{z\}$. Then, $|S' \cap S^*| = t - j$, so, by the induction assumption, S^* occurs exactly once as an induced t-subset. Now, $S^* \setminus \{y\}$ has cardinality $t - 1$, and is a member of $\mathcal{S}$. Hence, $S^* \setminus \{y\} \cup \{z\} = S''$ occurs exactly once as an induced t-subset in $\bigcup_{1 \le i \le m} \mathcal{A}_i(t)$.

So, we have proved that $\bigcup_{1 \le i \le m} \mathcal{A}_i$ is a $S(t, w, v)$. Each induced $(t - 1)$-subset occurs once in each $\mathcal{A}_i(t - 1)$, so each of $\mathcal{A}_1, \dots, \mathcal{A}_m$ is an $S(t - 1, w, v)$. This completes the proof.•

We will define an *optimal (t, w, v)-threshold scheme* to be a perfect (t, w)-threshold scheme having v shadows and $(v - t + 1) / (w - t + 1)$ keys. Unfortunately, there are not too many examples known of partitionable Steiner systems, so optimal threshold schemes are difficult to construct. We present two infinite classes in Section 3.

We finish this section by presenting an example of a well-known threshold system in this combinatorial setting. The scheme we discuss is the Shamir threshold scheme [7].

A prime number $p > w$ is chosen, and the key can be any integer $K \in \mathbf{Z}_p$ (so $m = p$). The set of shadows $X = \{(x, y) \in \mathbf{Z}_p \times \mathbf{Z}_p, 1 \le x \le w\}$ (so $v = pw$). Now, for every polynomial $h(x) \in \mathbf{Z}_p[x]$ having degree at most $t - 1$, we construct a block $B(h)$ as follows. The shadows in $B(h)$ are $(u, h(u))$, $1 \le u \le w$, and the key for $B(h)$ is $h(0)$. Hence, the number of blocks $b = p^t$.

This scheme is perfect. It is not difficult to see that any t-subset of shadows determine $h(x)$, and hence $K = h(0)$, uniquely, by means of Lagrange interpolation. Define a subset of shadows to be a *transversal* if no x-coordinate is repeated. Then is is easy to see, for each $\mathcal{A} = \phi^{-1}(K)$ $(K = 0, \dots, p - 1)$, that $\mathcal{A}(t - 1) = \{$every transversal of size $t - 1$, once each$\}$. Hence, no $(t - 1)$-subset gives any information as to the value of K.

It is also interesting to compare the number of keys to the bound on $m(t, w, v)$. The Shamir scheme has $v = pw$ and $m = p$, so $m = v / w$. The upper bound on m given in Corollary 2.3 is

$$(v - t + 1) / (w - t + 1).$$

Hence, for large values of v, the number of keys in the Shamir scheme is less than optimal by a factor of about

$$(w - t + 1) / w.$$

3. Some constructions for optimal threshold schemes

In this section, we present constructions for two classes of optimal threshold schemes with $t = 3$, and discuss their implementation. Our first construction is based on partitioning the set of all triples into Steiner triple systems.

Construction 1 Suppose $p \equiv 7 \bmod 8$ is a prime. Then there exists an optimal $(3, 3, p + 2)$ threshold scheme.

We describe a partition of the Steiner system $S(3, 3, p + 2)$ into p Steiner systems $S(2, 3, p + 2)$, due to R. Wilson [8]. Let $\mathbf{X} = GF(p) \cup \{\infty, \infty'\}$. Define $\mathcal{A}_0$ to consist of the following blocks:

one block: $\{\infty, \infty', 0\}$;
$(p^2 - 3p + 2) / 6$ blocks: $\{a, b, c\}$, where $a \neq b \neq c \neq a$, and $a + b + c = 0$;
$(p - 1) / 2$ blocks: $\{\infty, a, -2a\}$, where a is a quadratic residue in GF(p); and
$(p - 1) / 2$ blocks: $\{\infty', a, -2a\}$, where a is a quadratic non-residue in GF(p).

Then $\mathcal{A}_0$ is a $S(2, 3, p + 2)$. Define $\infty + i = \infty$ and $\infty' + i = \infty'$, for any $i \in GF(p)$. Now, for any $i \in GF(p)$, define

$$\mathcal{A}_i = \{x + i, y + i, z + i\} : \{x, y, z\} \in \mathcal{A}_0\}.$$

It is not difficult to show that $\bigcup_{0 \leq i < p} \mathcal{A}_i$ is an $S(3, 3, p + 2)$; hence we have the desired optimal threshold scheme.

Example: Suppose $p = 7$. Then $\mathcal{A}_0$ contains blocks:

$\{\infty, \infty', 0\}$, $\{0, 1, 6\}$, $\{0, 2, 5\}$, $\{0, 3, 4\}$, $\{1, 2, 4\}$, $\{3, 5, 6\}$, $\{\infty, 1, 5\}$, $\{\infty, 2, 3\}$, $\{\infty, 4, 6\}$, $\{\infty', 3, 1\}$, $\{\infty', 6, 2\}$, $\{\infty', 5, 4\}$.

Let's now briefly consider implementing such a scheme. We could take p to be some large prime, having 50 digits, for example. A key is any element i of GF(p). To determine shadows, it is necessary only to generate a random block of $\mathcal{A}_0$ and add i to each element. Observe that we can easily generate a random block of $\mathcal{A}_0$ from the recipe given above. It is unnecessary to construct any blocks ahead of time.

Given a block $\{x, y, z\}$, how is the key computed? Again, this is not difficult, if we consider the various possibilities. If $\{x, y, z\} = \{\infty, \infty', i\}$, then the key is i. If $\infty, \infty' \notin \{x, y, z\}$, then the key is $(x + y + z) / 3$. If the block is $\{\infty, x, y\}$, we proceed as follows. We know that $\{x, y\} = \{a + i, -2a + i\}$, where i is the key and a is a quadratic residue. Hence, $a = \pm (x - y) / 3$. Thus, we calculate $(x - y) / 3$, and test to see if it is a quadratic residue. We can do this easily in time $O(\log p)$. If $(x - y) / 3$ is a quadratic residue, then $a = (x - y) / 3$ and $i = x - a$; otherwise, $a = (y - x) / 3$ and $i = y - a$. Finally, if the block is $\{\infty', x, y\}$, the calculations are similar.

Hence, the key is calculated in time $O(1)$, unless the given block contains exactly one of ∞, ∞', in which case the calculation requires time $O(\log p)$. However, this second situation occurs with probability $O(1 / p)$, so on average, the calculation requires time $O(1)$. As well, we observe that the only arithmetic operations required (other than testing quadratic reciprocity) are a small number of addition or subtraction operations, and dividing by 3. The multiplicative inverse of 3 can be calculated ahead of time, so only a single multiplication operation would be required during key calculation.

We also observe that the number of keys, $v - 2$, is roughly three times the number of keys, $v / 3$, in the Shamir scheme when $t = w = 3$.

Our second construction is a partition of the planes of $AG(2m, 2)$ into 2-designs.

Construction 2 For every integer $m \geq 1$, there exists an optimal $(3, 4, 2^{2m})$ threshold scheme.

This is obtained by constructing a Steiner system $S(3, 4, 2^{2m})$ which can be partitioned into $2^{2m-1} - 1$ Steiner systems $S(2, 4, 2^{2m})$. This result is due to Baker [1].

Let $X = GF(2^{2m-1}) \times GF(2)$. Define $\mathcal{A}$ to consist of the planes of the affine geometry $AG(2m, 2)$, as follows:

blocks: $\{(a, i), (b, j), (c, k), (d, l)\}$ where $a + b + c + d = 0$ (in $GF(2^{2m-1})$) and $i + j + k + l = 0$ (in $GF(2)$).

Clearly, every block has the form $\{(a, 0), (b, 0), (c, 0), (d, 0)\}$, $\{(a, 1), (b, 1), (c, 1), (d, 1)\}$, or $\{(a, 0), (b, 0), (c, 1), (d, 1)\}$. Define a function $k: \mathcal{A} \rightarrow GF(2^{2m-1}) \setminus \{0\}$ as follows:

$$k(A) = (a^3 + b^3 + c^3 + d^3)^{1/3}, \text{ if } A = \{(a, 0), (b, 0), (c, 0), (d, 0)\},$$

$k(A) = (a^3 + b^3 + c^3 + d^3)^{1/3}$, if $A = \{(a, 1), (b, 1), (c, 1), (d, 1)\}$,
$k(A) = (a^3 + b^3 + c^3 + d^3 + (c + d)^3)^{1/3}$, if $A = \{(a, 0), (b, 0), (c, 1), (d, 1)\}$.

Then, define $\mathcal{A}_x = \{A\colon k(A) = x\}$ ($x \in GF(2^{2m-1}) \setminus \{0\}$). Then, it can be proved that each $\mathcal{A}_x$ is a $S(2, 4, 2^{2m})$ (see [1]).

Key calculation is accomplished as follows. We observe that, since $GCD(2^{2m-1} - 1, 3) = 1$, there exists a (unique) multiplicative inverse of 3 (mod $2^{2m-1} - 1$). Denote this number by t. Then, $x^{1/3} = x^t$, for any $x \in GF(2^{2m-1})$.

Let's also consider how to generate a random block in $\mathcal{A}_x$. This is made easier by the observation that, for any $\omega \in GF(2^{2m-1}) \setminus \{0\}$, $k(\omega A) = \omega \cdot k(A)$, where ωA is defined to be the block

$$\{(\omega a, i), (\omega b, j), (\omega c, k), (\omega d, l)\} \text{ (where } A = \{(a, i), (b, j), (c, k), (d, l)\}).$$

Hence, if we generate any random block $A \in \mathcal{A}$, we can then obtain a random block in $\mathcal{A}_x$ by multiplying A by $x \cdot k(A)^{-1}$.

In this scheme, the number of keys, $(v - 2) / 2$, is about 2 times the number of keys, $v / 4$, in the Shamir (3, 4)-scheme.

References

1. R. D. Baker, Partitioning the planes of $AG_{2m}(2)$ into 2-designs, Discrete Math. 15 (1976), 205-211.

2. A. Beutelspacher, Geometric structures as threshold schemes, preprint.

3. G. R. Blakely, Safeguarding cryptographic keys, Proc. N. C. C., vol. 48, AFIPS Conference Proceedings 48 (1979), 313-317.

4. G. R. Blakely and C. Meadows, Security of ramp schemes, Lecture Notes in Computer Science 196 (1985), 242-268.

5. I. Holyer, The NP-completeness of edge-colouring, SIAM J. Computing 10 (1981), 718-720.

6. S. C. Kothari, Generalized linear threshold scheme, Lecture Notes in Computer Science 196 (1985), 231-241.

7. A. Shamir, How to share a secret, Comm. of the ACM (22), 1979, 612-613.

8. R. M. Wilson, Some partitions of all triples into Steiner triple systems, Lecture Notes in Math. 411 (1974), 267-277.

A REALIZATION SCHEME FOR THE IDENTITY-BASED CRYPTOSYSTEM

Hatsukazu TANAKA
Department of Electrical Engineering
Kobe University
Rokko, Nada, Kobe 657, JAPAN

Abstract

At the Crypto'84, Shamir has presented a new concept of the identity-based cryptosystem, but no idea is presented on the realization scheme. In this paper a new realization scheme of the modified identity-based cryptosystem has been proposed. The basic idea of the scheme is based on the discrete logarithm problem and the difficulty of factoring a large integer composed of two large primes. The scheme seems to be very secure if all members of the system keep their secret keys safe, but if a constant number of users conspire, the center secret will be disclosed. Then it has a close relation to the well-known "threshold scheme". To cope with the conspiracy, the basic system is extended to get a new scheme of which "threshold" becomes higher. Detail considerations on the scheme are also given.

I. Introduction

At the Crypto '84, Shamir[1] has presented a new concept of the identity-based cryptosystems and signature schemes. He has proposed himself a realization scheme of the new concept of signature, but no idea is presented on a realization scheme of the identity-based cryptosystem. In this paper, modified it slightly without changing the basic important functions, a realization scheme is proposed. The basic idea of the scheme is based on the two well-known one-way functions, i.e. a factorization of a large integer composed of two large primes, and a discrete logarithm. The scheme is very simple, but it is possible to realize the Shamir's concept of the identity-based cryptosystem perfectly if all members of the system protect their secret informations safe. However, the original scheme has a crucial problem such that the center secret can be disclosed if some users conspire, because the scheme resembles to the well-known secret sharing system [2,3], i.e. a "threshold scheme". In order to overcome such a difficult problem, we extend the original scheme to a new one by introducing a new concept of "user's group" and "exchange", where the secret informations of two users are exchanged if their group numbers are different. As a result the threshold number of users necessary for conspiracy will be increased, and hence, the amount of calculations to raise the threshold can be decreased.

II. Identity-Based Cryptosystem

First we introduce the Shamir's original concept of identity-based cryptosystem which enables any pair of users to communicate securely, without exchanging private or public keys, without keeping any directories, and without using the services of a third party. The scheme assumes the existence of trusted key generation center, whose sole purpose is to give each user a personalized smart card when he first joins the network. The information embedded in this card enables the user to encrypt the messages he sends and to decrypt the messages he receives in a totally independent way, regardless of the other party. Previously issued cards do not have to be updated when new users join the network, and the various centers do not have to coordinate their activities or even to keep a user list. The centers can be closed after all the cards are issued, and the network can continue to function in a completely decentralized way for an infinite period. The block diagram of this concept is shown in Fig.1.

However, it seems to be very difficult to realize the original concept directly. Then with slight modification without changing the basic important functions of the cryptosystem, a new modified cryptosystem as shown in Fig.2 is obtained, and a realization scheme is proposed in the following Chapters.

III. A Realization Scheme

A)Basic System

Let p and q be two large primes and their product be $n=pq$ of which the Euler's totient function is given by $\Phi(n)=(p-1)(q-1)$. Let t be an arbitrary but not small positive integer, and let g be an integer which has a large period and satisfies $\max\{p,q\} < g < n$. Then select any t integers $x_\ell (1 \leq \ell \leq t)$ such that $\max\{p,q\} < x_\ell < \Phi(n)$, and assume that a user j's identity number is ID_j which is uniquely expanded to a large integer e_j using a one-way function f, i.e. $e_j=f(ID_j)$. Here the center calculates a set of the following t integers $S_{j\ell} (1 \leq \ell \leq t)$ less than n and sends it to j.

$$S_{j\ell} = g^{d_j x_\ell} \pmod{n}, \quad 1 \leq \ell \leq t \tag{1}$$

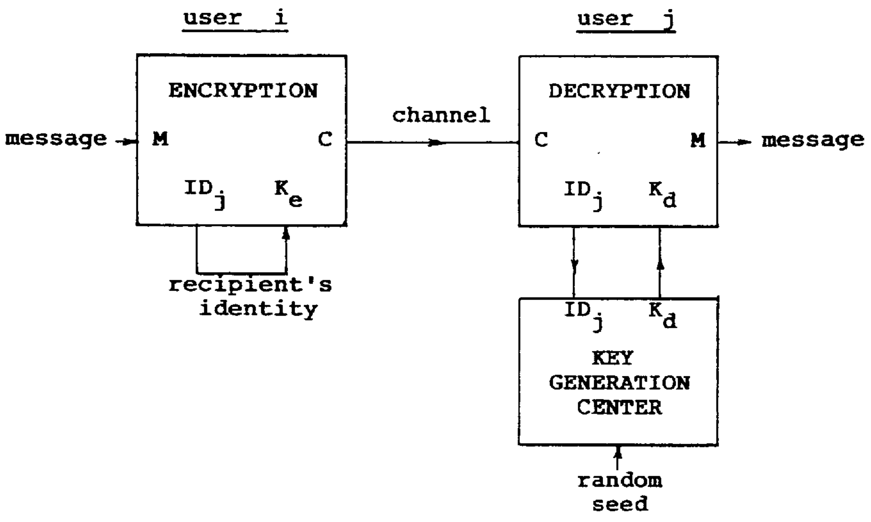


Fig.1 The Shamir's original identity-based cryptosystem.

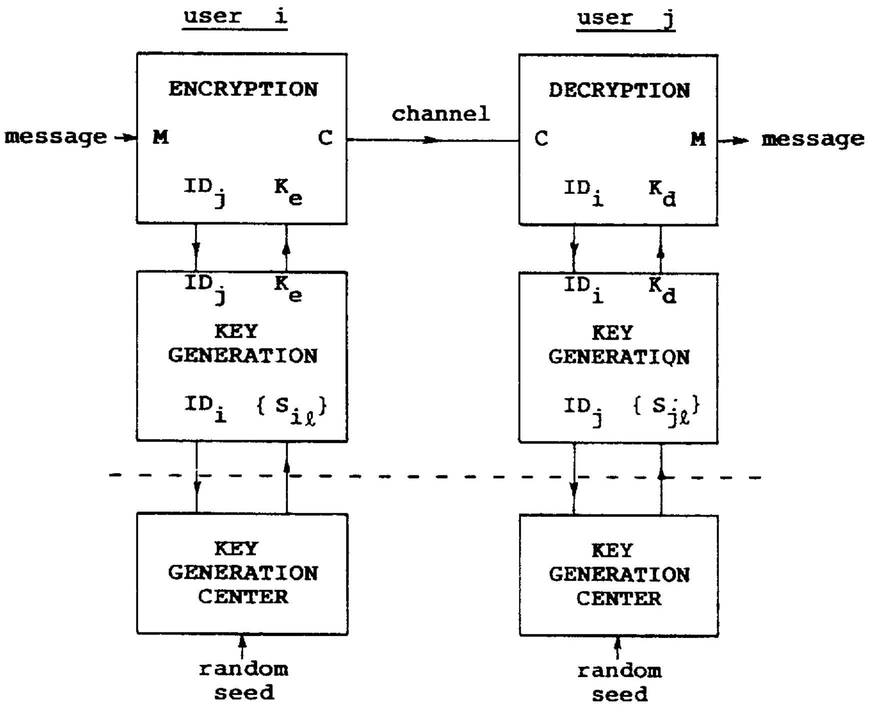


Fig.2 A modified identity-based cryptosystem.

where

$$f_{j\ell} = e_j^{\ell} \pmod{n} = \{f(ID_j)\}^{\ell} \pmod{n} \qquad (2)$$

and

$$d_j = \sum_{\ell=1}^{t} x_\ell f_{j\ell} \pmod{\Phi(n)} \qquad (3)$$

The system parameters are summarized as follows.

$$\begin{array}{ll} \text{Center} & \left\{ \begin{array}{ll} \text{Public-Key} & K_{CP} = \{ n,t,f \} \\ \text{Secret-Key} & K_{CS} = \{ p,q,g,x_\ell (1 \le \ell \le t) \} \end{array} \right. \\ \text{User } j & \left\{ \begin{array}{ll} \text{Public-Key} & K_{Pj} = \{ ID_j \} \\ \text{Secret-Key} & K_{Sj} = \{ S_{j\ell} (1 \le \ell \le t) \} \end{array} \right. \end{array}$$

B)Common-Key Generation

When a user i wants to generate a common-key with a user j, i calculates

$$K_{ij}^{(i)} = \prod_{\ell=1}^{t} S_{i\ell}^{f_{j\ell}} \pmod{n} \qquad (4)$$

using his secret key $K_{Si} = \{ S_{i\ell} \ (1 \le \ell \le t) \}$ and j's public key $K_{Pj} = \{ID_j\}$, where

$$f_{j\ell} = \{f(ID_j)\}^{\ell} \pmod{n}, \quad 1 \le \ell \le t \qquad (5)$$

And when a user j wants to generate a common-key with a user i, j calculates

$$K_{ji}^{(j)} = \prod_{\ell=1}^{t} S_{j\ell}^{f_{i\ell}} \pmod{n} \qquad (6)$$

using his secret key $K_{Sj} = \{S_{j\ell} \ (1 \le \ell \le t)\}$ and i's public key $K_{Pi} = \{ ID_i\}$, where

$$f_{i\ell} = \{f(ID_i)\}^{\ell} \pmod{n}, \quad 1 \le \ell \le t. \qquad (7)$$

Here we must show $K_{ij}^{(i)} = K_{ji}^{(j)}$ so that two users i and j may succeed to obtain a common-key.

Theorem 1. $k_{ij}^{(i)} = k_{ji}^{(j)} = g^{d_i d_j} \pmod{n}$ (8)

Proof: We will examine that the same expression can be derived by transforming $k_{ij}^{(i)}$ and $k_{ji}^{(j)}$ using Eqs.(1),(2)and (3).

$$k_{ij}^{(i)} = \prod_{\ell=1}^{t} S_{i\ell}^{f_{j\ell}} \pmod{n} = \prod_{\ell=1}^{t} g^{d_i x_\ell f_{j\ell}} \pmod{n}$$

$$= g^{d_i \sum_{\ell=1}^{t} x_\ell f_{j\ell}} \pmod{n} = g^{d_i d_j} \pmod{n},$$

and

$$k_{ji}^{(j)} = \prod_{\ell=1}^{t} s_{j\ell}^{f_{i\ell}} \pmod{n} = \prod_{\ell=1}^{t} g^{d_j x_\ell f_{i\ell}} \pmod{n}$$

$$= g^{d_j \sum_{\ell=1}^{t} x_\ell f_{i\ell}} \pmod{n} = g^{d_i d_j} \pmod{n}.$$

Hence,

$$k_{ij}^{(i)} = k_{ji}^{(j)} = g^{d_i d_j} \pmod{n}. \qquad \text{(Q.E.D.)}$$

From the above theorem, we denote the common-key with

$$k_{ij} = g^{d_i d_j} \pmod{n}. \qquad (9)$$

Remark : If a user j generates a common-key with himself using his public-key, he can obtain a key

$$k_{jj} = g^{d_j^2} \pmod{n}. \qquad (10)$$

which has an important application to encipher his private database or to generate a conference key.

C) Enciphering and Deciphering

Once a common key between a pair of users i and j is generated, the enciphering and deciphering can be performed using the well-known algorithm of common-key cryptosystem such as DES[4] or FEAL[5].

IV. Considerations on Security

The realization scheme proposed above seems to be secure if all members of the system protect their secret keys safe. However, if a number of users conspire, the center secret will be disclosed. Then it is very important for us to consider the algorithm to extract the center secret and the number of users who should join a conspiracy for success. Concerning it the following two theorems are established, though their proofs will be given later in the full paper.

Theorem 2. If the number of users who join a conspiracy is less than t, the center secret can not be disclosed. That is, the number of users T must satisfy $T \geq t$ in order to succeed in any conspiracy.

Theorem 3. When a factorization of n=pq or its equivalent information is given, the center secret can be disclosed if T=t users conspire.

It is very interesting to notice that the above theorem clearly shows that the center secret is delivered according to the well-known "threshold scheme" with a threshold value t. Therefore it is desirable to select t as large as possible from the point of security.

Generally, a factorization of n is unknown because two primes p and q are the center secret. Under this condition the following conjecture is established on the number of users to succeed for obtaining the center secret by a user's conspiracy.

Conjecture. When a factorization of n=pq or its equivalent information is not known, the center secret can not be disclosed if the number of users who join a conspiracy is less than t+1 for $t \geq 1$, i.e. the threshold value for success in a conspiracy is t+1.

It is desirable to select t as large as possible to protect the center secret from the user's conspiracy, but the necessary amount of memory capacity to store the users secret $S_{j\ell}$ $(1 \leq \ell \leq t)$ is $t\lceil \log_2 n \rceil$ (bits), where $\lceil\ \rceil$ shows the ceiling function, and increases as t becomes larger. Hence, the maximum number of possible users to keep the network system secure for any user's conspiracy and the amount of necessary memory capacity are exchanged.

V. Extension of the Basic System

In order to increase the number of users to join a network of the identity-based cryptosystem proposed above, it is necessary to select a parameter t large enough to prohibit the user's conspiracy. However, when t increases by a factor M, the number of necessary computations to generate a common-key also increases by a factor M.

In this Chapter we extend the basic system to get a new scheme which can keep our system secure against the user's conspiracy by pulling up the "threshold". The scheme introduces a new concept of "user's group" determined uniquely by the user's identity number ID, and exchanges the user's secret information between any two users in the different groups with no interaction and without leakage of any knowledge on their secret informations.

Let M be the number of user's groups and c be any positive integer. Then the group number N $(0 \leq N \leq M-1)$ of a user j is determined by

$$N = \{e_j\}^c \pmod M = \{f(ID_j)\}^c \pmod M. \qquad (11)$$

Here, introducing a new one-way function $\theta(N)$ of N, g is changed to

$$g_N = g^{\theta(N)} \pmod n \qquad (12)$$

and Mt integers $x_\ell^{(N)}$ $(1 \leq \ell \leq t,\ 0 \leq N \leq M-1)$ are selected as

$$\max\{p,q\} \leq x_\ell^{(N)} = y_\ell z_\ell^{(N)} \leq \Phi(n), \qquad (13)$$

where y_ℓ is a measure of $\Phi(n)$, and $z_\ell^{(N)}$ is an integer which satisfies

$\gcd\{z_\ell^{(N)},\Phi(n)\}=1$. Then the secret information of a user j in the group N_j is given by

$$S_{j\ell}^{(N_j)} = g_{N_j}^{d_j^{(N_j)} x_\ell^{(N_j)}} \pmod{n}, \quad 1 \le \ell \le t \tag{14}$$

where

$$f_{j\ell} = e_j^\ell \pmod{n} = \{f(ID_j)\}^\ell \pmod{n} \tag{15}$$

and

$$d_j^{(N_j)} = \sum_{\ell=1}^{t} x_\ell^{(N_j)} f_{j\ell} \pmod{\Phi(n)}. \tag{16}$$

When a user j in the group N_j wants to get a common-key with a user i in the group N_i, j's secret information $S_{j\ell}^{(N_j)}$ is exchanged with i's secret information by

$$S_{j\ell}^{(N_jN_i)} = \left\{S_{j\ell}^{(N_j)}\right\}^{\delta_\ell^{(N_i)}} \pmod{n}, \tag{17}$$

where the exchange information $\delta_\ell^{(N_i)}$ is given by

$$\delta_\ell^{(N_i)} = \frac{\beta\, x_\ell^{(N_i)}\, \mathrm{LCM}\{\,\theta(N_j),\, \theta(N_i)\}}{x_\ell^{(N_j)}\ \theta(N_j)} \pmod{\Phi(n)} \tag{18}$$

and β is a random integer, and then the common-key between them is obtained by

$$k_{ji}^{(j)} = \prod_{\ell=1}^{t} \left\{S_{j\ell}^{(N_jN_i)}\right\}^{f_{i\ell}} \pmod{n}$$

$$= \prod_{\ell=1}^{t} \left\{S_{j\ell}^{(N_j)}\right\}^{\delta_\ell^{(N_i)} f_{i\ell}} \pmod{n}, \tag{19}$$

where the secret informations necessary for the exchange $\{\delta_\ell^{(N)};\ 1 \le \ell \le t,\ 0 \le N \le M-1\}$ except $\delta_\ell^{(N^*)}$ for the user's own N^* is calculated by the center and delivered to all users beforehand being accompanied with their own secret information. Following the same process, a user i can obtain the common-key with j by

$$k_{ij}^{(i)} = \prod_{\ell=1}^{t} \left\{S_{i\ell}^{(N_iN_j)}\right\}^{f_{j\ell}} \pmod{n}$$

$$= \prod_{\ell=1}^{t} \left\{S_{i\ell}^{(N_i)}\right\}^{\delta_\ell^{(N_j)} f_{j\ell}} \pmod{n}. \tag{20}$$

Theorem 4.

$$k_{ij}^{(i)} = k_{ji}^{(j)} = g^{\beta \, \mathrm{LCM}\{\theta(N_i),\theta(N_j)\} \, d_i^{(N_i)} d_j^{(N_j)}} \pmod n \quad (21)$$

Proof: The theorem can be proved by deriving the same expression from $k_{ij}^{(i)}$ and $k_{ji}^{(j)}$ shown by Eqs.(19) and (20).

$$k_{ji}^{(j)} = \prod_{\ell=1}^{t} \left\{ S_{j\ell}^{(N_j)} \right\}^{\delta_\ell^{(N_i)} f_{i\ell}} \pmod n$$

$$= \prod_{\ell=1}^{t} g^{\theta(N_j) d_j^{(N_j)} x_\ell^{(N_j)} \frac{\beta \, x_\ell^{(N_i)} \mathrm{LCM}\{\theta(N_j),\theta(N_i)\}}{x_\ell^{(N_j)} \theta(N_j)} f_{i\ell}} \pmod n$$

$$= g^{\beta \, \mathrm{LCM}\{\theta(N_j),\theta(N_i)\} d_j^{(N_j)} \sum_{\ell=1}^{t} x_\ell^{(N_i)} f_{i\ell}} \pmod n$$

$$= g^{\beta \, \mathrm{LCM}\{\theta(N_j),\theta(N_i)\} d_j^{(N_j)} d_i^{(N_i)}} \pmod n,$$

and

$$k_{ij}^{(i)} = \prod_{\ell=1}^{t} \left\{ S_{i\ell}^{(N_i)} \right\}^{\delta_\ell^{(N_j)} f_{j\ell}} \pmod n$$

$$= \prod_{\ell=1}^{t} g^{\theta(N_i) d_i^{(N_j)} x_\ell^{(N_i)} \frac{\beta \, x_\ell^{(N_j)} \mathrm{LCM}\{\theta(N_i),\theta(N_j)\}}{x_\ell^{(N_i)} \theta(N_i)} f_{j\ell}} \pmod n$$

$$= g^{\beta \, \mathrm{LCM}\{\theta(N_i),\theta(N_j)\} d_i^{(N_i)} \sum_{\ell=1}^{t} x_\ell^{(N_j)} f_{j\ell}} \pmod n$$

$$= g^{\beta \, \mathrm{LCM}\{\theta(N_i),\theta(N_j)\} d_i^{(N_i)} d_j^{(N_j)}} \pmod n.$$

Hence,

$$k_{ij}^{(i)} = k_{ji}^{(j)} = g^{\beta \, \mathrm{LCM}\{\theta(N_i),\theta(N_j)\} d_i^{(N_i)} d_j^{(N_j)}} \pmod n.$$

(Q.E.D.)

From the above theorem the common-key between two users i and j is given by

$$k_{ij} = g^{\beta \, \mathrm{LCM}\{\theta(N_i),\theta(N_j)\} d_i^{(N_i)} d_j^{(N_j)}} \pmod n. \quad (22)$$

Here it is very interesting to consider a special case when M=1 and β=1. The expression of Eq.(22) can be rewritten as

$$k_{ij} = g_0^{d_i^{(0)} d_j^{(0)}} \quad (\text{mod } n), \qquad (23)$$

which is equivalent to Eq.(9). Hence the latter scheme is an extension of the former basic system.

VI. Considerations on the Extended System

A) The Number of Necessary Computations

Let us assume that the maximum number of users to join the identiy-based cryptosystem is less than T=Mt and the user's conspiracy must not succeed even when a factorization of n=pq is possible. Then we compare the number of necessary computations to generate a common-key for the basic system with that of the extended system under the condition that the user's memory capacity is equal to T=Mt for the both systems.

For the basic system, t should be increased to T. Then T exponentations mod n and 2(T-1) multiplications mod n are necessary to excute the calculations for Eqs.(4)and (5). On the other hand it is necessary for the extended system only to excute 2t exponentations mod n and 2·(t-1) multiplications mod n. Then, as $M \geq 2$, the number of necessary computations for the extended system is much less than that of the basic system, especially when M is large.

B) Existence of $\{x_\ell^{(N_j)}\}^{-1}$ (mod $\Phi(n)$) in Eq.(18)

It is clear that there exists $\{x_\ell^{(N_j)}\}^{-1}$ (mod $\Phi(n)$) if $x^{(N_j)}$ and $\Phi(n)$ are relatively prime, but regretfully, such a condition is not satisfied because $\gcd\{x_\ell^{(N_j)}, \Phi(n)\} = y_\ell$. However, as the numerator of Eq.(18) includes $x_\ell^{(N_j)} = y_\ell z_\ell^{(N_j)}$, the greatest common divisor y_ℓ can be cancelled, and hence, $\{x_\ell^{(N_j)}\}^{-1}$ (mod $\Phi(n)$) exists.

Remark : *It is clear that a factor* $\theta(N_j)$ *of the denominator is a* divisor of LCM$\{\theta(N_i), \theta(N_j)\}$.

C) Probability That the Number of Users in a Group Is Greater Than or Equal to t

In the extended system a group number is specified by $N = \{f(ID)\}^c$ (mod n), and if more than or equal to t users belong to a group, the users conspiracy will become possible. Then in order to cope with the possibility, we must consider the probability that the number of users who belong to a specified group N is greater than or equal to t.

Let L be the number of all users in the system, then the probability is given by

$$P_r(L;M,t) = \sum_{i=t}^{L} \binom{L}{i}\left(\frac{1}{M}\right)^i\left(1-\frac{1}{M}\right)^{L-i}, \qquad (24)$$

where M is the number of groups and $\{f(ID)\}^c$ (mod M) is assumed to be a uniformly distributed random number in $[0,M-1]$. Some numerical data are given in Table I.

Table I.

$t = 100 \quad M = 100$

L	Pr(L; M,t)
1000	6.62×10^{-67}
2000	2.45×10^{-39}
3000	2.66×10^{-25}
4000	1.14×10^{-16}
5000	5.80×10^{-11}
6000	4.34×10^{-07}
7000	1.83×10^{-04}
8000	9.72×10^{-03}
9000	1.13×10^{-01}

$t = 200 \quad M = 200$

L	Pr(L; M,t)
4000	$(< 10^{-100})$
8000	1.31×10^{-74}
12000	2.56×10^{-47}
16000	1.55×10^{-30}
20000	1.75×10^{-19}
24000	4.97×10^{-12}
28000	4.78×10^{-07}
32000	7.53×10^{-04}
36000	5.61×10^{-02}

References

[1] A.Shamir, " Identity-based cryptosystems and signature schemes," Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196, Springer Verlarg, 1985.

[2] A.Shamir,"How to share a secret," Comm. ACM, vol.22, No.11, Nov. 1979.

[3] R.J.McEliece and D.V.Sarwate, "On sharing secrets and Reed-Solomon codes," Comm. ACM, vol.24, No.9, Sept. 1981.

[4] Data Encryption Standard. Federal Information Processing Standards (IFIPS) Publication No.46, National Bureau of Standards, January 1977.

[5] S.Miyaguchi,"Criteria for the strength of encipherment and standard-ization for cryptographic techniques," Proceedings of the 1986 Symposium on Cryptography and Information Security, Feb. 1986.

Equivalence Between Two Flavours of Oblivious Transfers

Claude Crépeau[†]

Laboratory for Computer Science
M.I.T.
545 Technology Square
Cambridge Massachusetts 02139 USA

1. INTRODUCTION

The concept of oblivious transfer (O.T.) that was introduced by Halpern and Rabin [HR] turned out to be a very useful tool in designing cryptographic protocols. The related notion of "one-out-of-two oblivious transfer" was proposed by Even, Goldreich and Lempel in [EGL] together with some applications. Some more applications of this protocol can be found in recent papers [BCR], [GMW]. So far, the two notions where believed to be closely related but not known to be equivalent. This paper presents a proof that these two notions are computationally equivalent.

Essentially, we show a protocol for "one-out-of-two oblivious transfer", based on the existence of a protocol for the oblivious transfer problem. The reduction presented does not depend on any cryptographic assumption and works independently of the implementation of O.T.. The implications of this reduction are:

-there exists a protocol for ANDOS [BCR] if and only if there exists a protocol for O.T.
-the completeness theorem of [GMW] can be based on the existence of O.T.

2. DEFINITIONS

Let us first remind the reader the flavours of O.T. we are considering. The concept of oblivious transfer (O.T.) was first introduced by Halpern and Rabin in [HR]. Essentially the O.T. is a two-party protocol such that:

Definition 1: (O.T.)

-Alice knows one bit b.
-Bob gets bit b from Alice with probability $\frac{1}{2}$.
-Bob knows whether he got b or not.
-Alice does not know whether Bob got b or not.

The related notion is the "one-out-of-two oblivious transfer" defined by Even, Goldreich and Lempel in [EGL]. This other protocol is:

[†] Supported in part by an NSERC postgraduate scholarship.

Definition 2: (one-out-of-two O.T.)

-Alice knows two bits b_0 and b_1.
-Bob gets bit b_k and not $b_{\bar{k}}$ with $Pr(k=0) = Pr(k=1) = \frac{1}{2}$
-Bob knows which of b_0 or b_1 he got.
-Alice does not know which b_k Bob got.

In both these cases, the outcome of the transfer cannot be forced or influenced by either Alice or Bob. Although the structure of these protocols is extremely similar, so far nobody had proven their equivalence. Since the fact that O.T. can be achieved from one-out-of-two O.T. is trivial, the problem essentially is to show how to achieve one-out-of-two O.T. from O.T..

3. PROTOCOL

Before going into the explanation of the protocol, let us introduce a generalization of the O.T. protocol in the following way and consider the general case instead of the specific case. We define the p-O.T. to be a protocol such that:

Definition 3: (p-O.T.)

-Alice knows one bit b.
-Bob gets bit b from Alice with probability p.
-Bob knows whether he got b or not.
-Alice does not know whether Bob got b or not.

3.1. General idea

The general idea of the protocol is to use the p-O.T. protocol many times over random bits until it is very likely that it worked roughly pn times. The trick is to choose n large enough so that the p-O.T. protocol works at least $\frac{2}{3}pn$ of the time and not more than $\frac{4}{3}pn$ of the time. Then to get a bit, two disjoint subsets of size $\frac{2}{3}pn$ will be used, one of which will contain only indices of some p-O.T. that worked and the other will necessarily contain some indices of p-O.T. that did not work. Then the bits of each subset will be XORed together with one of the two bits to be disclosed.

3.2. Details of the protocol

Assume Alice owns b_0, b_1 two secret bits. To disclose one of them to Bob without knowing which one Bob gets, they can do the following for $p \leq \frac{3}{4}$:

Protocol for one-out-of-two O.T.

Alice and Bob agree on a security parameter s.
Alice chooses at random Ks bits $r_1, r_2, \cdots, r_{Ks}$ for some constant K to be later determined.
For each of these Ks bits Alice uses the p-O.T. protocol to disclose the bit r_i to Bob
    with probability p.

Bob selects $U=\{i_1,i_2,\cdots,i_{\alpha_s}\}$ and $V=\{i_{\alpha_s+1},i_{\alpha_s+2},\cdots,i_{2\alpha_s}\}$ where $\alpha_s=\left\lceil \frac{2Kps}{3} \right\rceil$

with $U \cap V=\phi$ and such that he knows r_{i_j} for each $i_j \in U$.

Bob sends $(X,Y)=(U,V)$ or $(X,Y)=(V,U)$ to Alice at random.

Alice computes $m_0=\bigoplus_{x\in X} r_x$ and $m_1=\bigoplus_{y\in Y} r_y$.

Alice returns to Bob k, $b_k \oplus m_0$ and $b_{\bar{k}} \oplus m_1$ for a random bit k.

Bob computes $\bigoplus_{u\in U} r_u \in \{m_0,m_1\}$ and uses it to get his secret bit.

If we have $p>\frac{3}{4}$ then they use the protocol for $p=\frac{3}{4}$ with a different value of K as suggested below.

4. ANALYSIS

We claim the following result about this protocol:

Theorem:

For an appropriately chosen constant K,

Pr(Bob gets at least one of b_0, b_1)$\geq 1-2^{-s}$ and Pr(Bob gets more than one of b_0, b_1)$\leq 2^{-s}$.

Proof:

Assume first that $p\leq\frac{3}{4}$. Name x_i the random variable such that

$$x_i=\begin{cases} 0 & \text{if Bob did not get } r_i \\ 1 & \text{if Bob did get } r_i \end{cases}$$

First notice that by definition $Pr(x_i=1)=1-Pr(x_i=0)=p$. Consider the random variable $X_i=\sum_{j=1}^{i} x_j$. Since the x_i's are independent random variables, then X_i is distributed with a binomial distribution. According to Bernshtein's Law of Large Numbers [Kr]

$$Pr(\left|\frac{X_i}{i}-p\right|\geq\varepsilon)\leq 2e^{-i\varepsilon^2}$$

for every ε such that $0<\varepsilon\leq p(1-p)$. In particular if we set $i=Ks$ and $\varepsilon=\frac{p}{4}$ we get

$$0<\varepsilon\leq p(1-p)$$

because $p\leq\frac{3}{4}$ and also we get

$$Pr(\left|\frac{X_{Ks}}{Ks}-p\right|\geq\frac{p}{4})\leq 2e^{-\frac{Ksp^2}{16}}\leq 2^{-s}$$

for $K\geq\frac{12}{p^2}$.

However, what we really are interrested in is

Pr(Bob gets at least one of b_0, b_1) and Pr(Bob gets more than one of b_0, b_1)

But we have that

$$\begin{aligned} &Pr(\text{Bob gets at least one of } b_0, b_1) \\ &=1-Pr(\text{Bob gets none of } b_0, b_1) \\ &=1-Pr(X_{Ks}<\left\lceil \frac{2Kps}{3} \right\rceil) \end{aligned}$$

$$=1-Pr\left(p-\frac{X_{Ks}}{Ks}>\frac{p}{3}+\frac{\frac{2Kps}{3}-\left\lceil\frac{2Kps}{3}\right\rceil}{Ks}\right)$$
$$\geq 1-Pr\left(p-\frac{X_{Ks}}{Ks}>\frac{p}{3}-\frac{1}{Ks}\right)$$

Since $s\geq 1, K\geq\frac{12}{p^2}$ and $p^2\leq p$ we get,

$$\geq 1-Pr\left(p-\frac{X_{Ks}}{Ks}>\frac{p}{3}-\frac{p}{12}\right)$$
$$\geq 1-Pr\left(p-\frac{X_{Ks}}{Ks}>\frac{p}{4}\right)$$
$$\geq 1-Pr\left(\left|\frac{X_{Ks}}{Ks}-p\right|\geq\frac{p}{4}\right)$$
$$\geq 1-2^{-s}$$

and

$$Pr(\text{Bob gets more than one of } b_0, b_1)$$
$$=Pr\left(X_{Ks}\geq 2\left\lceil\frac{2Kps}{3}\right\rceil\right)$$
$$\leq Pr\left(X_{Ks}\geq 2\frac{2Kps}{3}\right)$$
$$=Pr\left(\frac{X_{Ks}}{Ks}-p\geq\frac{p}{3}\right)$$
$$\leq Pr\left(\left|\frac{X_{Ks}}{Ks}-p\right|\geq\frac{p}{3}\right)$$
$$\leq Pr\left(\left|\frac{X_{Ks}}{Ks}-p\right|\geq\frac{p}{4}\right)$$
$$\leq 2^{-s}$$

QED.

Now, let's see the case $p>\frac{3}{4}$.

$$Pr(\text{Bob gets at least one of } b_0, b_1)$$
$$\geq Pr(\text{Bob gets at least one of } b_0, b_1 \mid p=\tfrac{3}{4})$$
$$\geq 1-2^{-s}$$

whenever $K\geq\frac{64}{3}$. And also

$$Pr(\text{Bob gets more than one of } b_0, b_1)$$
$$=Pr(X_{Ks}=Ks)$$
$$=p^{Ks}$$
$$\leq 2^{-s}$$

for $K\geq\frac{1}{lg\frac{1}{p}}$. So $K\geq max\left(\frac{64}{3},\frac{1}{lg\frac{1}{p}}\right)$ is a sufficient condition for our purpose.

QED.

Essentially, this theorem is claiming that Bob will get one of the bits except with an exponentially small probability, and Alice knows that he cannot get more than one of them except also with an exponentially small probability. In other words, this protocol achieves the one-out-of-two O.T. requirements with probability $1-2^{-s}$.

5. APPLICATIONS

In [BCR] one can find a reduction between a problem named AN2BP (All or Nothing 2 Bits Problem) and a very general disclosure problem: ANDOS (All or Nothing Disclosure of Secrets). Essentially AN2BP is identical to one-out-of-two O.T. except that Bob choses the random bit k used by Alice to decide which bit he gets. So the protocol we describe above accomplishes AN2BP if k is supplied by Bob. This reduction leads to the conclusion that ANDOS can be achieved from any p-O.T., for any constant p. Some more generalizations of O.T. can also be used as basis for reductions and will be explored in a further paper.

In [GMW], a completeness theorem for interactive protocol is presented based on the existence of one-way functions and one-out-of-two O.T. protocols. This completeness theorem can now be based on the existence of p-O.T. and one-way functions. It seems possible that p-O.T. is easier to construct directly than one-out-of-two O.T., in general.

6. OPEN PROBLEMS

An interresting problem is to transform an O.T. in which Alice learns with probability q whether Bob got the bit b or not, or an O.T. in which Bob always learns a bias about b into a one-out-of-two O.T.. Also it would be very interresting to find a way of achieving one of these variations only using one-way functions.

7. ACKNOWLEDGEMENTS

I wish to thank Gilles Brassard, Joe Kilian and Silvio Micali for the discussions we had about the protocol. I want to thank Ernie Brickell for proof reading this version of the paper.

8. REFERENCES

[BCR] Brassard G., Crépeau C. and Robert J.-M., ''Information Theoretic Reductions Among Disclosure Problems'', *Proceedings of the 27th Annual IEEE Symposium on the Foundations of Computer Science*, 1986, pp. 168-173.

[EGL] Even S., Goldreich O. and Lempel A., ''A randomized Protocol for Signing Contracts'', *Communications of the ACM*, Vol. 28, No. 6, 1985, pp. 637-647.

[GMW] Goldreich O., Micali S. and Wigderson A., ''How To Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority'', *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing*, 1987, pp. 218-229.

[HR] Halpern J. and Rabin M.O., ''A Logic to Reason about likehood'', *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing*, 1983, pp. 310-319.

[Kr] Kranakis, E., ''Primality and Cryptography'', John Wiley & Sons, 1986.

A construction for authentication / secrecy codes from certain combinatorial designs

D. R. Stinson
Department of Computer Science
University of Manitoba

Abstract

If we agree to use one of v possible messages to communicate one of k possible source states, then an opponent can successfully impersonate a transmitter with probability at least k / v, and can successfully substitute a message with a fraudulent one with probability at least (k - 1) / (v - 1). We wish to limit an opponent to these bounds. In addition, we desire that the observation of any two messages in the communication channel will give an opponent no clue as to the two source states. We describe a construction for a code which achieves these goals, and which does so with the minimum possible number of encoding rules (namely, $v \cdot (v - 1) / 2$). The construction uses a structure from combinatorial design theory known as a perpendicular array.

1. Authentication and secrecy

In this paper, we study the properties of codes with respect to secrecy and authentication. We are interested in the *unconditional*, or *theoretical*, security provided by such codes. That is, we assume that any opponents have unlimited computational resources. The theory of unconditional secrecy is due to Shannon [12]. More recently, Simmons has developed an analogous theory of unconditional authentication.

We shall use the model of authentication theory as described by Simmons in [13], [14], and [15]. In this model, there are three participants: a transmitter, a receiver, and an opponent. The *transmitter* wants to communicate some information to the *receiver*, whereas the *opponent* wants to deceive the receiver. The opponent can either impersonate the receiver, making him accept a fradulent message as authentic; or, modify a message which has been sent by the transmitter.

More formally, we have a set of k source states S, a set of v messages M, and a set of b encoding rules E. A *source state* $s \in S$ is the information that the transmitter wishes to communicate to the receiver. The transmitter and reciever will have secretly chosen an *encoding rule* $e \in E$

beforehand. An encoding rule e will be used to determine the ***message*** e(s) to be sent to communicate any source state s. It is possible that more than one message can be used to determine a particular source state (this is called *splitting*). However, in order for the receiver to be able to uniquely determine the source state from the message sent, there can be at most one source state which is encoded by any given message $m \in M$ (i.e. $e(s) \neq e(s')$ if $s \neq s'$).

We are interested in the security of such a code with respect to both *secrecy* and *authentication*. Suppose an opponent observes i distinct messages being sent over the communications channel (where $i \geq 0$). He knows that the same key is being used to transmit the i messages, but he does not know what that key is. If we consider the code as a secrecy system, then we make the assumption that the opponent can only observe the messages being sent. Our goal is that the opponent be unable to determine any information regarding the i source states from the i messages he has observed.

In [8] and [11], the following scenario for authentication is investigated. As before, an opponent observes i distinct messages. The opponent then sends a message m' to the receiver, hoping to have it accepted as authentic (this message m' must be distinct from the i messages already sent). In [8], Massey calls this a *spoofing attack* of order i. We remark that the special cases $i = 0$ and $i = 1$ have been studied extensively by Simmons and other people (see [1], [13], [14], [15], and [16]). The case $i = 0$ is called the *impersonation* game, and the case $i = 1$ is called the *substitution* game.

For any i, there will be a probability distribution on the set of i source states which occur. We ignore the order in which the i source states occur, and assume that no source state occurs more than once. Also, we assume that *any* set of i source states has a non-zero probability of occurring. Given a set of i source states S, we define p(S) to be the probability that the source states in S occur.

Given the probability distributions on the source states described above, the receiver and transmitter will choose a probability distribution for E, called an *encoding strategy*. If splitting occurs, then they will also determine a *splitting strategy* to determine $m \in M$, given $s \in S$ and $e \in E$ (this corresponds to non-deterministic encoding).

Once the transmitter / receiver have chosen encoding and splitting strategies, we can define for each $i \geq 0$ a probability denoted Pd_i, which is the probability that the opponent can deceive the transmitter / receiver with a spoofing attack of order i.

In this paper, we consider only codes without splitting. We shall use the following notation. Given any encoding rule e, we define $M(e) = \{e(s): s \in S\}$, i.e. the set of messages permitted by encoding rule e. For a set M' of distinct messages, and an encoding rule e, define $f_e(M') = \{s: e(s) \in M'\}$, i.e. the set of source states which will be encoded under encoding rule e by a message in M'. As well, for a set M' of distinct messages, define $E(M') = \{e \in E: M' \subseteq M(e)\}$, i.e. the set of encoding rules under which all the messages in M' are permitted. It is useful to think of a code as being represented by a b x k matrix, where the rows are indexed by encoding rules, the columns are indexed by source states, and the entry in row e and column s is e(s).

Theorem 1 [8, p. 12]. In an authentication system without splitting,

$$Pd_i \geq \frac{k-i}{v-i}.$$

Proof: Suppose the opponent observes the i messages in the set $M' = \{m_1, \ldots m_i\}$ in the channel. For $m \in M \setminus M'$, let payoff(M', m) denote the probability that message m would be accepted as authentic. Then we have

$$\text{payoff}(m, M') = \frac{\sum_{e \in E(M' \cup \{m\})} p(e) \cdot p(S = f_e(M'))}{\sum_{e \in E(M')} p(e) \cdot p(S = f_e(M'))}.$$

It is not difficult to calculate

$$\sum_{m \in M \setminus M'} \text{payoff}(m, M') = k - i.$$

Hence, there exists some $m \in M' \setminus M$ such that payoff(m, M') $\geq$ (k - i) / (v - i). For every set M' of i source states, the opponent can choose such an m. This proves that $Pd_i \geq (k - i) / (v - i)$.•

Following Massey [8], we say that the authentication system is *L-fold secure against spoofing* if $Pd_i = (k - i) / (v - i)$ for $0 \leq i \leq L$.

When we consider the secrecy properties of a code, we desire that no information be conveyed by the observation of the messages which are transmitted. We say that a code has *perfect L-fold secrecy* if, for every set M_1 of at most L messages observed in the channel, and for every set S_1 of at most $|M_1|$ source states, we have $p(S_1 \mid M_1) = p(S_1)$. That is, observing a set of at most L messages in the channel does not help the opponent determine the L source states.

The purpose of this note is to give a simple construction for a system which achieves perfect 2-fold secrecy and is 1-fold secure against spoofing, and does so with the minimum possible number of keys, as given by the following bound.

Theorem 2 If a code achieves perfect L-fold secrecy and is (L - 1)-fold secure against spoofing, then

$$b \geq \binom{v}{L}.$$

Proof: Let M_1 be a set of $i \leq L - 1$ messages which are permitted under a particular encoding rule. Let x be any message not in M_1. Suppose there is no encoding rule under which all messages in $M_1 \cup \{x\}$ are valid. Then a suitable modification of the proof of Theorem 1 shows that we would have $Pd_i > (v - i) / (k - i)$, a contradiction. Hence, it follows that every L-subset of messages is valid under at least one encoding rule.

Now, pick any L-subset of messages M_2. In order to achieve perfect L-fold secrecy, the messages in M_2 must encode every possible L-subset of source states. Hence, M_2 is a valid set of messages under at least $\binom{k}{L}$ encoding rules. Now, if we count L-subsets of messages, we get

$$b \cdot \binom{k}{L} \geq \cdot \binom{v}{L} \cdot \binom{k}{L}.$$

This completes the proof.•

In the remainder of the paper, we shall study the existence of codes where the bound of Theorem 2 is met with equality. Hence, we define on *optimal L-code* to be a code which achieves perfect L-fold secrecy, is (L - 1)-fold secure against spoofing, and has precisely $\binom{v}{L}$ encoding rules.

We give a construction for optimal 2-codes in Section 2. We remark that a construction for codes which provide perfect 1-fold secrecy and are 1-fold secure against spoofing was given in [16].

2. Constructions for optimal 2-codes using perpendicular arrays

Our interest is in constructing optimal L-codes. We can do this for $L = 2$ using a type of combinatorial design known as a perpendicular array. These arrays were first studied in [9], and have been investigated by several researchers in combinatorial design theory since then (see [5], [6] and [7]). A *perpendicular array* PA(n, k) is a $v \cdot (v - 1) / 2$ by k array, A, of the symbols $\{1, \dots, v\}$, which satisfies the following property:

> for any two columns i and j of A, and for any two distinct symbols $x, y \in \{1, \dots, v\}$, there is a unique row r such that $\{A(r, i), A(r, j)\} = \{x, y\}$.

We have the following construction using perpendicular arrays.

Theorem 3 If there exists a PA(v, k), where $k > 2$, then there is a code for k source states with v messages and $v \cdot (v - 1) / 2$ encoding rules, which achieves perfect 2-fold secrecy and is 0-fold secure against spoofing.

Proof: We construct an encoding rule from each row r of the perpendicular array A: for each row $r = (x_1, \dots, x_k)$ of A, we define an encoding rule $e_r(s) = (x_s\colon 1 \le s \le k)$. We shall use each encoding rule with probability $2 / (v \cdot (v - 1))$.

Let's first verify that $Pd_0 = k / v$. This follows immediately from the following easily proved property of perpendicular arrays: if $k > 2$, then every symbol occurs exactly $(v - 1) / 2$ times in each column of a PA(v, k).

Next, we check that we have perfect 2-fold secrecy. Again, this is an almost immediate consequence of the definition of perpendicular array. Given any two messages m and m', and given any two source states s and s', there is exactly one encoding rule e such that $\{e(s), e(s')\} = \{m, m'\}$. Hence, we have $p(S = \{s, s'\} \mid \{m, m'\}) = p(S = \{s, s'\})$.•

The following example illustrates that a code constructed by means of Theorem 2 will not necessarily be 1-fold secure against spoofing.

Example 1 The following is a PA(5, 3):

0	1	2
1	2	3
2	3	4
3	4	0
4	0	1
0	3	1
1	4	2
2	0	3
3	1	4
4	2	0

Suppose the source probability distribution is (p_1, p_2, p_3), where $p_1 > p_2 > p_3$. What happens if the opponent observes the message 0 being transmitted? The conditional probability distribution on the encoding rules, given that message 0 is observed, is:

$$\left(\frac{p_1}{2}, 0, 0, \frac{p_3}{2}, \frac{p_2}{2}, \frac{p_1}{2}, 0, \frac{p_2}{2}, 0, \frac{p_3}{2}\right).$$

If the opponent substitutes message 0 with message 1, it will be accepted as authentic with probability

$$\frac{p_1}{2} + \frac{p_2}{2} + \frac{p_1}{2}$$

$$= \frac{1}{2} + \frac{p_1 - p_3}{2}.$$

In fact, the opponent's optimal substitution strategy is to replace any message m by the message (m + 1) mod 5. This yields

$$Pd_1 = \frac{1}{2} + \frac{p_1 - p_3}{2}.$$

A special type of perpendicular array will allow us to attain $Pd_1 = (k - 1) / (v - 1)$. A PA(v, k) A is said to be *cyclic* (and is denoted CPA(v, k)) if any cyclic permutation of the columns of A yields an array which can be obtained from A by means of a suitable permutation of the rows of A. That is, if $(x_1, \dots, x_k)$ is a row of A, then $(x_2, \dots, x_k, x_1)$ is also a row of A.

Example 2 A cyclic PA(5, 5).

0	1	2	3	4
1	2	3	4	0
2	3	4	0	1
3	4	0	1	2
4	0	1	2	3
0	2	4	3	1
1	3	0	4	2
2	4	1	0	3
3	0	2	1	4
4	1	3	2	0

We have the following

Theorem 4 If there exists a cyclic PA(v, k), then there is an optimal 2-code for k source states with v messages.

Proof: Let A be a CPA(v, k). Construct the code as in Theorem 2. We need only verify that $Pd_1 = (k - 1) / (v - 1)$. Let m and m' be two distinct messages. We have

$$\text{payoff}(m, m') = \frac{\sum_{e \in E(m, m')} p(e) \cdot p(S = f_e(m))}{\sum_{e \in E(m')} p(e) \cdot p(S = f_e(m))}$$

$$= \frac{\sum_{e \in E(m, m')} p(S = f_e(m))}{\sum_{e \in E(m')} p(S = f_e(m))}$$

$$= \frac{\sum_{e \in E(m, m')} p(S = f_e(m))}{\frac{v-1}{2}}.$$

Now, there are k·(k - 1) / 2 rows r of A for which m, m' occur in row r. For each source state j, there are exactly (k - 1) / 2 encoding rules e_r where m, m' occur in row r and $e_r(j) = m$. Then,

$$\sum_{e \in E(m, m')} p(S = f_e(m)) = \frac{k-1}{2}.$$

Hence, payoff(m, m') = (k - 1) / (v - 1), as desired. This is true for any two messages m, m'. Hence, no matter what the opponent's substitution strategy, he will deceive the receiver with this probability.

Cyclic perpendicular arrays have been the subject of several recent papers, such as [2], [5] and [6]. It is not difficult to see that the existence of a CPA(v, k) requires that v and k be odd, and that $2k \mid v \cdot (v - 1)$. For k = 3 and 5, these conditions are necessary and sufficient for existence, with one exception.

Theorem 5

1) ([6]) A CPA(v, 3) exists if and only if $v \equiv 1$ or 3 modulo 6.
2) ([5]) A CPA(v, 5) exists if and only if $v \equiv 1$ or 5 modulo 10, $v \neq 15$.

For k > 5, only sporadic results are known. One class of cyclic PAs is given by

Theorem 6 ([2]) k is odd and $v \equiv 1$ modulo 2k is a prime power, then there is a CPA(v, k).

We will discuss the construction of Theorem 6 in some detail in Section 3, but let's first observe that, although the existence of a CPA(v, k) is sufficient for the existence of an optimal 2-code, it is not necessary. We shall say that a PA(v, k) is *pair-column balanced* if, for every pair of symbols x, y, the following property is satisfied:

Among the rows containing x and y, x and y each occur (k - 1) / 2 times in each column.

Lemma 7 If there exists a pair-column balanced PA(v, k), then there exists an optimal 2-code for k source states with v messages.

Proof: The proof is identical to that of Theorem 4.•

Example 3 (van Rees [17]) A pair-column balanced PA(11, 3). Develop the following 5 rows modulo 11:

0	1	2
0	9	7
0	3	6
0	4	8
0	5	10

We have the following result concerning pair-column balanced PA(v, 3).

Theorem 8 There exists a pair-column balanced PA(v, 3) for all odd $v \geq 3$, $v \neq 5, 17$. Further, there does not exist a pair-column balanced PA(5, 3).

Proof: This result follows easily from the following recursive pairwise balanced design construction for PAs ([7, Lemma 4.1]). Let v be a positive integer, and let $K \subseteq \{2, \ldots, v-1\}$. A (v, K)-PBD (*pairwise balanced design*) is a set X of v elements (points) and a set **B** of subsets of X (blocks), such that every (unordered) pair of points occurs in a unique block $B \in \mathbf{B}$, and $|B| \in K$ for every $B \in \mathbf{B}$. Suppose we have a (v, K)-PBD, and for every $n \in K$, there exists a PA(n, k). Then we can construct a PA(v, k), by taking a PA(|B|, k) on symbol set B, for every $B \in \mathbf{B}$. It is easy to check if every "input" PA(|B|, k) is pair-column balanced, then so is the resulting PA(v, k).

Now, we already have that there is a pair-column balanced PA(n, 3) for every $n \equiv 1$ or 3 mod 6, and for n = 11. In [4, Theorem 3.3], it is shown that there exists a (v, {3, 11})-PBD for all $v \equiv 5$ modulo 6, $v \geq 23$. The above construction produces pair-column balanced PA(v, 3) for all these values of v. There remain only v = 5 and v = 17 to consider. An exhaustive search ([17]) has shown that no pair-column balanced PA(5, 3) exists. The case v = 17 remains open.•

Apparently, no examples of pair-column balanced PA(v, 5) are known, other than the cyclic PAs.

3. Implementing optimal 2-codes

In this section, we describe the construction and implementation of the optimal 2-codes guaranteed by Theorem 6.

Suppose k is odd and $v \equiv 1$ modulo 2k is a prime power. The perpendicular array PA(v, k) is constructed as follows ([6]). Let ω be a primitive element in the finite field GF(v), and let $\alpha = \omega^{(n-1)/k}$. For each $i = 1, \dots, (v-1)/2k$, for each $j = 0, \dots, k-1$, and for each $\beta \in GF(v)$, define a row

$$\beta + \omega^i\alpha^j \qquad \beta + \omega^i\alpha^{1+j} \qquad \beta + \omega^i\alpha^{2+j} \qquad \dots \qquad \beta + \omega^i\alpha^{k-1+j}$$

This defines $v \cdot (v-1)/2$ rows, which is the right number, at least. It is not difficult to see that the resulting array is indeed a PA(v, k). Also, it is clearly cyclic: the rows obtained by varying j, for fixed i and β, are all cyclic shifts.

Hence, we can pick a random encoding rule e by generating a random 3-tuple (i, j, β) of the form described above. A source state s $(0 \le s \le k-1)$ would then be encoded as

$$e(s) = \beta + \omega^i\alpha^{s+j}.$$

Also, given an encoded message m, we can solve for the source state s by means of the equation

$$\alpha^{s+j} = (m - \beta) / \omega^i.$$

Observe that this requires the calculation of a logarithm in the finite field GF(v). This is made easier by the knowledge that $0 \le s \le k-1$. As well, if v is a prime, and $v - 1$ has only small prime factors, then an algorithm of Pohlig and Hellman ([10]) can be used which has computational complexity $O((\log v)^2)$.

Finally, let's consider the sizes of messages and encoding rules, as a function of the authentication security. Recall that we have $Pd_0 = k/v$ and $Pd_1 = (k-1)/(v-1)$, where we have k source states, m messages, and $v \cdot (v-1)/2$ encoding rules. Since we want Pd_0 and Pd_1 to be small, we might consider taking $v = k \cdot (k-1) + 1$ (assuming, of course, that it is a prime power). In this case, we have $Pd_0 \approx 1 / (k - 1)$ and $Pd_1 = 1 / k$. We require roughly $2 \cdot \log_2 k$ message bits in order to transmit $\log_2 k$ bits of source. If we send two messages, then we transmit $2 \cdot \log_2 k$ bits of source with $4 \cdot \log_2 k$ bits of message, with perfect secrecy. The encoding rule requires about $4 \cdot \log_2 k$ bits.

It is interesting to compare these space requirements with that of the well-known "one-time pad". The one-time pad achieves perfect secrecy by requiring one bit of key (analogous to our encoding rules) for every bit of source. It is also well-known that this much key is required if perfect secrecy is to be attained. In the example we have constructed above, we have 2 bits of "key" for every bit of source communicated (when two messages are sent). If we send only one message, then we have 4 bits of "key" for every bit of source, but we gain a high degree of security with respect to authentication. It bears repeating that these results cannot be achieved with less "key" (Theorem 2).

References

1. Ernest F. Brickell, A few results in message authentication, Congressus Numerantium 43 (1984), 141-154.
2. A. Granville, A. Moisiadis and R. Rees, Nested Steiner n-gon systems and perpendicular arrays, preprint.
3. E. Gilbert, F. J. MacWilliams and N. J. A. Sloane, Codes which detect deception, Bell System Tech. J. 53 (1974), 405-424.
4. C. Huang, E. Mendelsohn and A. Rosa, On partially resolvable t-partitions, Annals Disc. Math. 12 (1983), 169-183.
5. C. C. Lindner and D. R. Stinson, Steiner pentagon systems, Discrete Math. 52 (1984), 67-74.
6. C. C. Lindner and D. R. Stinson, The spectrum for the conjugate invariant subgroups of perpendicular arrays, Ars Combinatoria 18 (1984), 51-60.
7. C. C. Lindner, R. C. Mullin and G. H. J. van Rees, Separable orthogonal arrays, Utilitas Math., to appear.
8. J. L. Massey, Cryptography - A selective survey, in "Digital Communications" (1986), 3-21.
9. R. C. Mullin, P. J. Schellenberg, G. H. J. van Rees and S. A. Vanstone, On the construction of perpendicular arrays, Utilitas Math. 18 (1980), 141-160.

10. S. C. Pohlig and M. E. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, IEEE Trans. on Inform. Theory 24 (1978), 106-110.

11. P. Schobi, Perfect authentication systems for data sources with arbitrary statistics, preprint.

12. C. E. Shannon, Communication theory of secrecy systems, Bell System Tech. J. 28 (1949), 656-715.

13. Gustavus J. Simmons, A game theory model of digital message authentication, Congressus Numerantium 34 (1982), 413-424.

14. Gustavus J. Simmons, Message Authentication: A game on hypergraphs, Congressus Numerantium 45 (1984), 161-192.

15. Gustavus J. Simmons, Authentication theory / Coding theory, in "Advances in Cryptology: Proceedings of CRYPTO 84", Lecture Notes in Computer Science, vol. 196, 411-432, Springer Verlag, Berlin, 1985.

16. D. R. Stinson, Some constructions and bounds for authentication codes, J. of Cryptology, to appear.

17. G. H. J. van Rees, private communication.

A DIGITAL SIGNATURE BASED ON A CONVENTIONAL ENCRYPTION FUNCTION

by Ralph C. Merkle
Elxsi
2334 Lundy Place
San Jose, CA 95131

ABSTRACT

A new digital signature based only on a conventional encryption function (such as DES) is described which is as secure as the underlying encryption function -- the security does not depend on the difficulty of factoring and the high computational costs of modular arithmetic are avoided. The signature system can sign an unlimited number of messages, and the signature size increases logarithmically as a function of the number of messages signed. Signature size in a 'typical' system might range from a few hundred bytes to a few kilobytes, and generation of a signature might require a few hundred to a few thousand computations of the underlying conventional encryption function.

INTRODUCTION

While digital signature systems have been proposed[1,3,5,10] that rely only on conventional encryption functions (or on one-way functions) none has quite succeeded in providing the convenience of systems based on more complex mathematical problems such as factoring[2,4]. A significant reason for interest in systems whose security is based only on one-way functions is that the existence of such functions seems assured, while the complexity of factoring and the most efficient factoring algorithm are still open questions of great interest. This is not an issue of purely academic interest, especially in light of the large number of 'unbreakable' cryptographic systems that were subsequently broken.

A second advantage is the reduced computational cost as compared with systems that require modular arithmetic: a software implementation of DES (the Data Encryption Standard) runs much faster than exponentiation modulo N, so a digital signature system based on use of DES would likewise benefit. This savings becomes more significant in a hardware implementation because DES chips are already available at low cost from many manufacturers, and are already present in many existing systems. The new digital signature system is very fast indeed when retro-fitted to a system that already has a DES chip (or a hardware implementation of any conventional encryption function).

To make this article self-contained we first briefly review some previously known results on one-way functions and one-time signatures, and then show how a one-time signature system can be used in a new way to provide a digital signature system that overcomes the limitations and drawbacks that have hampered the acceptance and use of this approach.

ONE WAY FUNCTIONS

A one-way function F is a function that is easy to compute, but difficult to invert. Given x, computing y=F(x) is easy. Given y, determining any x such that F(x)=y is hard.

One way functions can be based on conventional encryption functions by observing that deducing the key given the plain text and cipher text is very hard. If we define a conventional encryption function as: S_{key}(plaintext) = ciphertext, then we can define a one way function F(x)=y by computing $S_x(0)=y$. That is, we encrypt a constant using x as the key. The resulting ciphertext is the output of the one way function. Deducing x given that we know y is now equivalent to determining the key given that we know the plaintext is 0 and the ciphertext is y.

One way hash functions -- e.g., a one way function which accepts an arbitrarily large input (say, a few kilobytes) and produces a small fixed size output (say, 64 bits) -- can be based on repeated applications of a conventional encryption function in a similar way. The design of one way hash functions should be approached with caution: the most obvious approaches are sometimes vulnerable to 'square root' attacks. For example, if we wish to reduce 112 bits to 64 bits using DES, the obvious technique is to break the 112 bits into two 56-bit blocks and then double encrypt a fixed constant. That is, if the two 56-bit blocks are designated K1 and K2, then compute: $S_{K2}(S_{K1}(0))$. Unfortunately, it is possible to determine a new K1' and K2' in about 2^{28} operations that will produce the same result using a 'meet in the middle' or 'square root' attack. While such attacks can be avoided, it is important to know that they exist and must be guarded against.

We assume that a secure one way hash function is available, possibly based on some conventional encryption function. We shall denote this function F.

SIGNING A ONE-BIT MESSAGE

This section and the next section provide a brief introduction to one-time signatures for those readers unfamiliar with them. These two sections can be skipped without loss of continuity.

Person A can sign a one-bit message for B by using the following protocol: first, in a pre-computation A uses F to one-way encrypt two values of x: x[1] and x[2] -- producing two values of y: y[1] and y[2]. Second, A makes y[1] and y[2] public while keeping x[1] and x[2] secret. Finally, if the one-bit message is a '1', A signs it by giving x[1] to B. If the one-bit message is a '0', A signs it by giving x[2] to B.

If the one-bit message was '1', B can prove that A signed it by presenting x[1] and showing that F(x[1])=y[1]. Because F and the y's are public, anyone can verify the results of this computation. Because only A could know x[1] and x[2], B's knowledge of x[1] implies that A gave x[1] to B -- an act which, by prior agreement, means that A signed the message '1'.

SIGNING A SEVERAL BIT MESSAGE

If A generated many x's and many y's, then A could sign a message with many bits in it. This is the Lamport-Diffie one-time signature[1]. To sign an n-bit message requires 2n x's and 2n y's.

Merkle[3] proposed an improvement to this method which reduces the signature size by almost two-fold. Instead of generating two x's and two y's for each bit of the message, A generates only one x and one y for each bit of the message to be signed. When one of the bits in the message to be signed is a '1', A releases the corresponding value of x -- but when the bit to be signed is a '0', A releases nothing. Because this allows B to pretend that he did not receive some of the x's, and therefore to pretend that some of the '1' bits in the signed message were '0', A must also sign a count of the '0' bits in the message. Now, when B pretends that a '1' bit was actually a '0' bit, B must also increase the value of the count field -- which can't be done. Because the count field has only $\log_2$ n bits in it, the signature size is decreased by almost a factor of two -- from 2n to n+$\log_2$ n.

As an example, if we wished to sign the 8-bit message '0100 1110' we would first count the number of '0' bits (there are 4) and then append a 3-bit count field (with the value 4) to the original

8-bit message producing the 11-bit message '0100 1110 100' which we would sign by releasing x[2], x[5], x[6], x[7] and x[9]. B cannot pretend that he did not receive x[2], because the resulting erroneous message -- '0000 1110 100' would have 5 0's in it, not 4. Similarly, pretending he did not recieve x[9] would produce the erroneous message '0100 1110 000' in which the count field indicates that there should be no 0's at all. There is no combination of x's that B could pretend not to have received that would let B concoct a legitimate message.

Winternitz[6] proposed an improvement which reduces the signature size by several fold. Instead of being able to sign a one-bit message by pre-computing y[1]=F(x[1]) and y[2]=F(x[2]), A would be able to sign a 2-bit message by pre-computing y[1]=F(F(F(F(x[1])))) and y[2]=F(F(F(F(x[2])))). Notationally, we will show repeated applications of the function F with a superscript -- $F^3(x)$ is F(F(F(x))), $F^2(x)$ is F(F(x)), $F^1(x)$ is F(x), and $F^0(x)$ is x. If A wishes to sign message m -- which must be one of the messages '0', '1', '2', or '3' -- then A reveals $F^m(x[1])$ and $F^{3-m}(x[2])$. B can easily verify the power of F that A used by counting how many more applications of F must be used to reach y. Computing complimentary powers of both x[1] and x[2] is necessary because B might pretend to have received a higher power than A actually sent him. That is, if A sent $F^2(x[1])$ to B, B could compute $F^3(x[1])$ and pretend that A had sent THIS value instead. However, if B does this then B must compute $F^0(x[2])$ as well -- which A would have computed and sent to B if A had actually signed the message '3'. Because A actually sent $F^1(x[2])$, this means B must compute x[2] given only F(x[2]) -- which he can't do. Sending the complimentary powers of x[1] and x[2] in this technique is directly analogous to releasing either x[1] or x[2] in the Lamport-Diffie method.

Though this example shows how to sign one of four messages, the system can be generalized to sign one of n messages by pre-computing $y[1]=F^{n-1}(x[1])$ and $y[2]=F^{n-1}(x[2])$.

The almost two-fold improvement proposed by Merkle for the 1-bit one-time signature generalizes to the Winternitz one-time signature.

Thus, the original one-time signature system proposed by Lamport and Diffie, and improved by Winternitz and Merkle, can be used to sign arbitrary messages and has excellent security. The storage and computational requirements for signing a single message are quite reasonable. Unfortunately, signing more messages requires many more x's and y's and therefore a very large entry in the public file (which holds the y's). To allow A to sign 1000 messages might require roughly 10,000 y's -- and if there were 1000 different users of the system, each of whom wanted to sign 1000 messages, this would increase the storage requirement for the public file to hundreds of megabytes -- which is unwieldy and has effectively prevented use of these systems.

AN INFINITE TREE OF ONE-TIME SIGNATURES

The general idea in the new system is to use an infinite tree of one-time signatures. For simplicity, we assume that the tree is binary. The root of the infinite tree is authenticated simply by placing it in the public file. Each node of the tree performs three functions: (1) it authenticates the left sub-node (2) it authenticates the right sub-node and (3) it signs a single message. Because there are an infinite number of nodes in the tree, an infinite number of messages can be signed. To perform these three functions, each node must have three signatures -- a 'left' signature, a 'right' signature, and a 'message' signature. The 'left' signature is used to 'sign off' on the left sub-node, the 'right' signature is used to 'sign off' on the right sub-node, while the 'message' signature is available to sign a user message.

Notationally, it is convenient to number the nodes in the tree in the following fashion:

The root node is designated '1'.
The left sub-node of node i is designated 2i.
The right sub-node of node i is designated 2i+1.

This assignment of numbers has many convenient properties. It uniquely numbers every node in the infinite tree; the left and right sub-nodes are easily computed from a parent node; and the parent node can be computed from the sub-node by a simple integer division by 2. Note that if we start from node 1 and follow the left sub-node at each node, the node numbers are: 1,2,4,8,16,32,64, ...

We adopt some further notational conventions to distinguish between the x's and y's used to sign different messages at different nodes in the tree -- in particular, we shall use a three-dimensional array of x's and y's: array x[<node number>, <left,right, or message>, <index within the one-time signature>]. If we use the original Lamport-Diffie method (involving 128 x's per signature) then all the x's at node i would be:

x[i,left,1], x[i,left,2] ... x[i,left,128]
x[i,right,1], x[i,right,2] ... x[i,right,128]
x[i,message,1], x[i,message,2] ... x[i,message,128]

We will designate all the x's for the 'left' signature at node i by x[i,left,*]. Similarly, we shall designate all the y's associated with the message signature at node i by y[i,right,*]. We shall designate all the x's at node i (left, right, and message) by x[i,*,*].

We shall often wish to apply a one way hash function to all the y's for a given signature, so we define the notation F(y[i,right,*]) to mean use of the one way hash function F applied to all the y's for the right signature of node i.

Thus, our fundamental data structures will be two infinite three-dimensional arrays x and y, where each y is computed from the corresponding x by applying F.

We shall often wish to compute a 'hash total' for all the y's at a given node. We do this by first applying F to each signature individually, and then applying F to the three resultant values. We define the function HASH(i) as:

HASH(i) = F(F(y[i,left,*]), F(y[i,right,*]), F(y[i,message,*]))

This is the one way hash total for node i. It has the important property that if we already know HASH(i) and someone sends us what they claim are the y[i,*,*] values we can confirm that they sent us the correct values (or show that they sent the wrong values) by re-computing the function. If the value of HASH(i) computed from the values sent to us matches the value that we already know, then we know we have received the correct y[i,*,*] values.

Prior to the signature protocol, A enters HASH(1) into the public file. This value authenticates the root node of A's authentication tree, and it is assumed that it is publicly known to everyone.

We can now describe the algorithm that A uses to sign message m with signature i, and that B uses to check the signature.

THE NEW SIGNATURE ALGORITHM

Both A and B agree in advance on the message M to be signed. A selects the node i that will be used to sign it.

1. A sends i and y[i,message,*] to B.
2. A signs message M by sending B the appropriate subset of x's in x[i,message,*].
3. B checks that the released subset of the x[i,message,*] correctly sign message M when checked against the y[i,message,*].

4. A sends F(y[i,left,*]), F(y[i,right,*]) and F(y[i,message,*]) to B.
5. A computes HASH(i). By definition, this is:
   F(F(y[i,left,*]), F(y[i,right,*]), F(y[i,message,*]))
6. If the value of i is 1, then B checks that the value of HASH(1) computed from the values A transmitted matches the entry HASH(1) in the public file, and the algorithm terminates.
7. If i is even,
   A sends y[i/2,left,*] to B.
   A signs HASH(i) by sending the correct subset of x[i/2,left,*].
   B computes HASH(i) and verifies that it was properly signed by checking the x's against the y[i/2,left,*].
8. If i is odd,
   A sends y[i/2,right,*] to B.
   A signs HASH(i) by sending the correct subset of x[i/2,right,*].
   B computes HASH(i) and verifies that it was properly signed by checking the x's against the y's in y[i/2,right,*].
9. Both A and B replace i by i/2 and proceed to step 4.

When the algorithm terminates, B has $\log_2$ i one-time signatures, one of which is the one-time signature for message M that B actually wanted, while each of the others verifies the correctness and validity of the next signature -- and the validity of the 'root' signature is attested to by the entry in the public file. Thus, this 'audit trail' of one-time signatures starts with HASH(1), proceeds to HASH(i), and finally terminates with the one-time signature for message M.

It should be clear that this 'meta-system' can utilize any one-time signature system, and that improvements in the one-time signature system will produce corresponding improvements in the meta-systems performance. There is no particular reason to believe that current one-time signature systems have reached a plateau of perfection, and so further research into one-time signature systems might well produce worthwhile performance improvements.

It should also be clear that the use of a binary tree is arbitrary -- it could just as easily be a K-ary tree, and probably will be in practice. A binary tree requires $\log_2$ i one-time signatures, while a K-ary tree requires only $\log_K$ i one-time signatures -- which generally results in a smaller over-all signature size for larger values of K. However, in a K-ary tree the computation of HASH(i) becomes:

```
F(
 F(y[i,first-sub-node,*]),
    F(y[i,second-sub-node,*]),
    F(y[i,third-sub-node,*]),
    .
    .
    .
    F(y[i,Kth-sub-node,*])
    F(y[i,message,*])
    )
```

The computation at each node takes longer because all the y's for all the K sub-nodes must be re-computed, and each node in the resulting signature requires more memory. Thus, the optimal value of K can't be too large -- or it will run afoul of these limitations.

The problem of minimizing the additional authentication information required within each node as the value of K increases is actually interesting in its own right. As described above, the information required as part of the signature at each node will increase linearly with K. This has been reduced to $\log_2$ K in the author's previous 'tree-signature' method[3,7 page 170]. Combining the two systems into a single hybrid seems quite appropriate and would allow rather large values of K to be used efficiently. The author's previous 'tree-signature' method is quite different in concept from the current method, though both use a tree. It is interesting to note that Goldwasser, Micali, and Rivest[8,9] also use a tree-like structure to provide desirable theoretical properties in a digital signature. Their signature system is '...based on the existence of a "claw-free" pair of [trapdoor] permutations' which they build by multiplying together two large primes (as in the RSA system).

Finally, some readers might object that the infinite three dimensional arrays x and y might be awkward for user A to store -- and so a compaction scheme seems appropriate. The array of y's is computed from the array of x's, and so the y's need not actually be stored. The array of x's is randomly chosen by A in any fashion that A desires. A might just as well generate the x's in a secure pseudo-random fashion. In particular, A can compute x[i,j,k] by concatenating i, j, and k and then encrypting this bit pattern with a conventional encryption function using a secret key: x[i,j,k] = $S_{\text{A's secret key}}$(<i,j,k>). If we were to use DES, A's secret key need only be 56 bits. Even after many of the x's had been made public (in the course of signing various messages) it would be impossible to determine A's secret key. The pairs (<i,j,k>, x[i,j,k]) are plaintext-ciphertext pairs and by definition the key of a conventional encryption function cannot be determined even if many such pairs are known.

In practice, all A need remember is a single secret key (of perhaps 56 bits) and a simple integer count (of perhaps 20 or 30 bits) to keep track of which node in the tree was used to sign the last message. If the computations are ordered correctly, generating a signature can be done in a very small memory (128 bytes of RAM is more than enough). Such small memories (and even smaller) often occur in low-cost high-volume applications, such as 'smart-cards' (credit cards with a built-in computer).

CONCLUSION

A digital signature system has been presented which is based solely on a conventional encryption function. The algorithms to sign and check signatures are rapid and require only a very small amount of memory. The size of the signatures grows as the logarithm of the number of messages signed. Signature size and memory requirements can be traded off against computational requirements.

REFERENCES

1. 'New Directions in Cryptography', IEEE Trans. on Information Theory, IT-22, 6(Nov. 1976),644-654

2. 'A method for obtaining digital signatures and public-key cryptosystems.' CACM 21,2, Feb. 1978 120-126

3. 'Secrecy, Authentication, and Public Key Systems', Ralph C. Merkle, UMI Research Press 1982.

4. 'How to Prove Yourself: Practical Solutions to Identification and Signature Problems', Amos Fiat and Adi Shamir, 1986.

5. 'Making the Digital Signature Legal -- and Safeguarded', S.M. Lipton, S.M. Matyas, Data Communications, Feb. 1978 41-52.

6. Private Communication, Robert Winternitz, 1980.

7. 'Cryptography and Data Security', by Dorothy E.R. Denning, Addison Wesley 1982.

8. 'A "Paradoxical" solution to the Signature Problem', by Shafi Goldwasser, Silvio Micali and Ronald L. Rivest, from the Symposium on the Foundations of Computer Science, 1984, page 441-448.

9. 'A Digital Signature Scheme Secure Against Adaptive Chosen Message Attack', by Shafi Goldwasser, Silvio Micali and Ronald L. Rivest, extended abstract, 1986.

10. 'Cryptography: a New Dimension in Computer Data Security', by Carl H. Meyer and Stephen M. Matyas, Wiley 1982.

How to Make Replicated Data Secure

Maurice P. Herlihy and J. D. Tygar
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Many distributed systems manage some form of long-lived data, such as files or data bases. The performance and fault-tolerance of such systems may be enhanced if the repositories for the data are physically distributed. Nevertheless, distribution makes security more difficult, since it may be difficult to ensure that each repository is physically secure, particularly if the number of repositories is large. This paper proposes new techniques for ensuring the security of long-lived, physically distributed data. These techniques adapt replication protocols for fault-tolerance to the more demanding requirements of security. For a given threshold value, one set of protocols ensures that an adversary cannot ascertain the state of a data object by observing the contents of fewer than a threshold of repositories. These protocols are cheap; the message traffic needed to tolerate a given number of compromised repositories is only slightly more than the message traffic needed to tolerate the same number of failures. A second set of protocols ensures that an object's state cannot be altered by an adversary who can modify the contents of fewer than a threshold of repositories. These protocols are more expensive; to tolerate t-1 compromised repositories, clients executing certain operations must communicate with t-1 additional sites.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order Numbers 4864 and 4976, monitored by the Air Force Avionics Laboratory under Contracts F33615-84-K-1520 and N00039-84-C-0467.

1. Introduction

A *distributed system* consists of a collection of computers, called sites, that are geographically distributed and connected by a communications network. Examples of distributed systems include distributed inventory control systems, banking systems, airline reservation systems, and campus-wide file systems. Many distributed systems manage some form of long-lived data, such as files or databases. In this paper, we propose new techniques for ensuring the security of long-lived data in distributed systems. Our notion of security encompasses two properties: (1) *secrecy*, ensuring that unauthorized parties cannot observe confidential data (e.g., by stealing a disk pack), and (2) *integrity*, ensuring that unauthorized parties cannot corrupt or modify valuable data (e.g., by doctoring a disk pack).

Physical distribution makes security more difficult. In a centralized system, perhaps the most straightforward approach to security is to keep long-lived data in a physically secure location (e.g., keep the disk drive in a locked room.) When repositories for data are physically distributed, however, it is more difficult to ensure that each one is physically secure. As the number of sites increases, so does the number of ways in which the secrecy and integrity of the data can be compromised.

A similar difficulty arises when attempting to guarantee a distributed system's *availability*: the larger the number of independently-failing components, the smaller the likelihood they will all be working simultaneously, and the smaller the likelihood the system will be accessible when needed. This well-known phenomenon is typically addressed by designing distributed systems to be *fault-tolerant*, that is, able to function correctly in the presence of some number of failures. In particular, the availability of long-lived data can be enhanced by storing the data redundantly at multiple sites, a technique commonly known as *replication* [7, 11].

In this paper, we consider how replication protocols originally proposed to enhance fault-tolerance can be adapted to the more demanding requirements of security. For a given *threshold* value t, we describe and analyze several encryption-based secrecy protocols that ensure that an adversary cannot ascertain the object's state by observing the contents of fewer than t repositories. We then extend these protocols to guarantee integrity, ensuring that the object's state cannot be altered by an adversary who can tamper with the contents of fewer than t repositories. To keep our presentation as straightforward as possible, we assume that a communications subsystem provides secure and authenticated communication using known protocols, e.g., [2, 14, 10, 8]. Here, we consider only attacks that bypass the communications subsystem, isolating some set of repositories and directly observing or modifying their data.

A natural basis for analyzing the costs of our protocols is to compare the cost of replication for availability (tolerating t failures) with the cost of replication for security (tolerating t compromised sites). We find that:

- *Secrecy is cheap.* We propose two alternative secrecy protocols, one based on private key encryption, and one based on public key encryption. The private key scheme has the advantage that encryption itself is fast, but a client may sometimes have to communicate with additional sites the first time it operates on an object. The public key scheme is more flexible, but known public key encryption algorithms are slower than their private key counterparts. We believe that the private key encryption method could be practical for real systems since the cost of security is dominated by the cost of replication.

- *Integrity is expensive.* To tolerate *t-1* compromised repositories, clients executing certain operations must communicate with *t-1* additional sites. We show that our protocol is optimal in the sense that any protocol that requires less communication is vulnerable to a simple "spoofing" attack.

A naive solution, at least for security, is simply to issue encryption keys to clients, storing only encrypted data at vulnerable repositories. A weakness of this approach is that it does not address key management. If the client keeps the key in non-volatile storage, then an adversary can compromise the security of the data simply by compromising the client's local storage. If the client keeps the key in volatile memory, a key management service is necessary both to initialize clients, and to redistribute new keys when the data is reencrypted. Care must be taken that the key management service itself does not become a security risk (what happens if a site belonging to the key management service becomes compromised?) or an availability bottleneck (what happens if the key management service's sites are down?). The protocols described in this paper address the security and availability aspects of key management by integrating key distribution directly into the replication protocols.

In Section 2 we define some terminology and we review two algorithms that form the basis for our protocols: quorum consensus replication and shared secrets. In Section 3 we describe a protocol employing private key encryption to preserve the secrecy of data against a passive adversary, and in Section 4 we describe an alternative secrecy protocol employing public key encryption. Section 5 discusses protocols for on-the-fly reencryption. In Section 6, we describe a protocol that preserves the integrity of the data against an active adversary, and Section 7 concludes with a survey of related work and a brief discussion.

2. Background

2.1. Terminology

We use two notions of cryptographic security in this paper. One notion, of *bit-security,* [3, 19], implies that given ciphertext, no processor with randomized polynomial resources can derive information about any given bit in the corresponding cleartext with certainty greater than $1/2+\varepsilon$ for any $\varepsilon > 0$. Given the current limits of complexity theory, we do not have a way of proving bit-security without making some assumption about the lower bound on an algorithmic problem. For example, it has been shown [1] that the RSA [20] cryptosystem is bit-secure under the assumption that taking k^{th} roots modulo pq, a product of two large primes, cannot be done in randomized polynomial time and that the Rabin [15] signature schemes is bit-secure if factorization is not in randomized polynomial time. (These assumptions are generally accepted by the academic computer science community.)

The second notion, of *perfect security*, implies that given ciphertext, no processor with unlimited resources can derive a probability distribution of the corresponding cleartext other than a uniform distribution. For example, Shamir's secret sharing method, described in Section 2.3, satisfies the requirements of perfect security; the range of secrets compatible with fewer than a threshold number of pieces ranges over the entire set of integers in the finite field. Clearly, a cryptographic method that has perfect security is also bit-secure. When we say that a ciphertext gives *no information* about the corresponding cleartext, we mean that it satisfies either of the above definitions of security.

2.2. Quorum Consensus Replication

A complete description of quorum consensus replication is given elsewhere [11]. For brevity, we give two informal examples: files and counters.

A replicated object is implemented by two kinds of modules: *repositories* and *front-ends*. Repositories provide long-term storage for the object's state, while front-ends carry out operations for clients. Different objects may have different sets of repositories. Because front-ends can be replicated to an arbitrary extent, perhaps placing one at each client's site, the availability of a replicated object is dominated by the availability of its repositories. A *quorum* for an operation is any set of repositories whose cooperation suffices to execute that operation. A *quorum assignment* associates each operation with a set of quorums. An operation's quorums determine its availability, and the constraints governing an object's quorum assignments determine the range of availability properties realizable by a replication method.

A replicated file [7] is represented as a collection of timestamped versions, where timestamps are generated by logical clocks [12]. To read a file, a front-end reads the version from a *read quorum* of repositories and chooses the version with the latest timestamp. To write the file, a front-end generates a new timestamp and records the newly timestamped version at a write quorum of repositories. A quorum assignment is correct if and only if each read quorum has a non-empty intersection with each write quorum.

File replication algorithms are unsuited for certain kinds of objects. Consider a *counter* object that assumes integer values. It provides three operations: *Inc()* increments the counter by one, *Dec()* decrements the counter by one, and *Value()* returns the counter's current value. Such a counter is used as part of a protocol for dynamic reconfiguration of replicated objects [11]. The counter is a replicated *reference count* that keeps track of the number of existing references to a particular replicated object. The counter is incremented whenever a new reference is created, and decremented whenever an existing reference is destroyed. The system periodically checks the counter's value, reclaiming storage for objects whose counters have reached zero.

R1	R2	R3
1:01 Inc()		1:01 Inc()
1:02 Inc()	1:02 Inc()	
	2:03 Dec()	2:03 Dec()

Figure 2-1: A Replicated Counter

As shown in Figure 2-1, a replicated counter is represented as a partially replicated log of entries, where an entry is a timestamped record of an Inc or Dec operation. It is convenient to split an operation's quorums into two parts: an *initial* quorum and a *final* quorum. To increment or decrement the counter, the front-end simply sends a timestamped entry to a final quorum for the operation. To read the counter's value, the front-end merges the logs from a final quorum for the operation, using the timestamps to discard duplicates. A quorum assignment is correct if and only if (1) each final Dec quorum intersects each initial Value quorum, and (2) each final Inc quorum intersects each final Value quorum. Many other examples of replicated typed objects appear elsewhere [11].

2.3. Shared Secrets

In this section, we give a brief overview of Shamir's *secret sharing* algorithm [22]. This algorithm transforms a cleartext value *v* into *n* encrypted pieces such that any *t* pieces determine the value of *v*, but an adversary in possession of *t-1* pieces has no information, in the sense defined in Section 2.1, about the value of *v*.

We do all computations with integer residues modulo a large prime *p*. Let the polynomial

$$h(x) = r_1x^{t-1} + r_2x^{t-2} + \ldots + r_{t-1}x + v$$

where the r_i are independent random integer residues. The pieces of the secret are the values *h(1), ..., h(n)*. The secret itself is *v=h(0)*. Since *t* distinct values of *h* determine a nonsingular linear system of *t* equations in *t* variables, Gaussian elimination will allow us to recover the values of the coefficients of *h* and hence *v*. On the other hand, since our operation is over a finite field, *t-1* values of *h* yield no information about the value of *v*.

Creation of secrets require evaluation of a *t-1*-th degree polynomial (time $O(t)$) and recovery of the original text requires solution of a small linear system (time $O(t^3)$). (We are assuming here a fixed *p*; the complexity of these operations is polynomial in the logarithm of *p*.) This compares very favorably with standard private key and public key encryption techniques [13].

3. Private Key Secure Quorum Consensus

The protocol described here protects the secrecy of the object against an adversary who can observe the contents of fewer than *t* repositories. It depends on the existence of a bit-secure *private key* encryption scheme, in which a single key *K* is used for both encryption and decryption. The encryption scheme must be *probabilistic* [9] to ensure that repeated instances of the same cleartext (e.g., *Inc* entries for a replicated counter) produce different ciphertext instances.

3.1. Overview

The private key Secure Quorum Consensus (SQC) protocol is implemented by three kinds of sites:

1. *Front Ends* interact directly with clients, but have only volatile memory.
2. *Repositories* communicate only with other processors and have a long-term store.
3. A *Dealer* communicates only with other processors and has a source of random bits. The stores his data exclusively in volatile memory. The dealer could be a separate processor, but in general the dealer can be any one of a set of processors -- for example, the dealer can be one of the front ends or repositories. If one dealer becomes inaccessible, another processor can take over as the dealer, thus the dealer need not be a performance or availability bottleneck.

The SQC protocol consists of three phases:

1. *Object initialization:* The dealer chooses a random key *K*, and uses the shared secret protocol to divide *K* into *n* pieces such that any *t* pieces suffice to reconstruct *K*. One piece is sent to each of the *n* repositories for the object, and all data stored at the repositories will be encrypted with *K*
2. *Front-end initialization:* When a front-end comes on-line, it ascertains *K* by reading the pieces from *t* out of *n* repositories. The front-end records *K* in a volatile cache, which must be reinitialized on crash recovery, or when the object is reencrypted.
3. *Operation Execution:* The front-end executes a modified quorum consensus protocol. In the first step, it reads the encrypted data from an initial quorum of repositories. In the second step, it decrypts the data using *K*, performs the operation, and encrypts the new data using *K*.

Recall that all messages are assumed to be secure and authenticated by the communications subsystem.

The definition of a bit-secure private key cryptosystem and Shamir's secret sharing algorithm immediately imply:

Property 1: An adversary who reads the contents of fewer than *t* repositories can derive no information about the object's state.

Nevertheless, notice that an adversary may still be able to derive useful information from "traffic analysis," such as observing when an object's value changes or has been reencrypted. For example, if versions or log entries are timestamped in cleartext, then the amount of decryption needed to execute an operation can be reduced, but an adversary may be able to deduce information about the number and frequency of updates.

3.2. Examples

This section illustrates SQC by three examples representing three different availability/security trade-offs.

1. Consider a file for which the efficiency and availability of Read and Write operations are equally important. Here, read quorums, write quorums, and thresholds all require a majority $\lceil (n+1)/2 \rceil$ of repositories. Preserving secrecy incurs no additional cost in availability since a front-end can register at a quorum for its first operation. Up to $\lceil (n-1)/2 \rceil$ repositories may fail without loss of availability, and equally many repositories may be observed by an adversary without disclosing the contents of the file.

2. If the efficiency and availability of Read is more important, Read quorums could be defined to encompass any one repository, and Write quorums all *n*. Here, a threshold value of one is possible, but not prudent. Instead, likely threshold values range between two and a majority, trading the efficiency and availability of registration against the security of the data. Preserving secrecy incurs a one-time penalty, since a unregistered front-end cannot read the file from a read quorum. Nevertheless, the registration cost is amortized over the lifetime of the front-end.

3. At the other extreme, consider a replicated reference counter. The availability of the Inc and Dec operations is likely to be more important than the availability of the Value operation, since timely creation or destruction of references is more important than asynchronous garbage collection. Inc and Dec quorums could be defined to encompass any one repository, and Value quorums all *n*. Here, too, sensible threshold values range between two and a majority, and registration incurs a one-time availability penalty amortized over the lifetime of the front-end.

Preserving secrecy need not reduce availability if the threshold value can be set equal to the object's smallest quorum. If the smallest quorum consists of a single site, as in Examples 2 and 3, then private key SQC imposes a one-time registration penalty for each front-end. Clearly, some such penalty is inherent in Example 2, because if an unregistered front-end can ascertain the file's value from the data residing at a single repository, so can an adversary. More surprisingly, the registration penalty in Example 3 can be eliminated, as we show in the next section.

4. Public Key Secure Quorum Consensus

In this section we describe a variant of SQC in which the bit-secure private key scheme is replaced by a bit-secure *public key* scheme [6]. Instead of a single key K, we use an *encryption key* K_E and a *decryption key* K_D, where K_D cannot be derived from K_E with polynomial resources, and vice-versa. Instead of a single threshold t, we use an *encryption threshold* t_E and a *decryption threshold* t_D. The dealer generates K_E and K_D, and uses the shared secret protocol to divide each one into n pieces such that any t_E pieces suffice to reconstruct K_E, and any t_D pieces suffice to reconstruct K_D. A front-end executing an operation uses K_D to decrypt the data it reads and K_E to encrypt the data it writes.

Because the encryption and decryption thresholds can be chosen independently, public key SQC is more flexible than its private key counterpart. For example, consider the replicated counter example from the previous section, in which quorums for Inc and Dec consist of any one repository, and a Value quorum consists of all n. By setting t_E to one and t_D to n, we enhance performance, availability, and security, eliminating the registration penalty, and permitting up to n-1 repositories to be compromised without disclosing the counter's value. If integrity is not a concern, there is no reason not to set t_E to one, effectively making K_E public. As discussed in Section 6, however, preserving integrity against an active adversary will require a higher value for t_E.

5. On-the-Fly Reencryption

To manage secure replicated data, it is important to be able to reencrypt data easily. A object might be reencrypted periodically to render useless any pieces of the key that have been inadvertently exposed, or an object might be reencrypted in response to suspicion that its current key has become compromised. In this section, we discuss how SQC supports reencryption. For brevity, we restrict our discussion here to files and private key SQC. We assume that the encrypted text includes enough internal redundancy (c.f., Section 6) that a front-end attempting to decrypt data with an out-of-date key will detect its mistake and reregister.

Consider a file replicated among n repositories having read quorums, write quorums, and thresholds of sizes r, w, and t.

> **Property 2:** A front-end that knows K can reencrypt the object with key K' as long as it has access to $\max(r,w,n\text{-}t\text{+}1)$ repositories.

The front-end requires access to a read quorum to read the file's current value, and write quorum to write out the reencrypted value. Suppose there are t-1 inaccessible repositories $R_1,\ldots,R_{t-1}$ whose pieces of the key are $s_1,\ldots,s_{t-1}$. The goal of the reencryption protocol is to choose a new key value K' which splits into pieces $s_1', \ldots ,s_n'$ such that $s_i = s_i'$, for i between 1 and t-1. The front-end knows the values of the polynomial coefficients K, r_1, ..., r_{t-1}, from which it can derive the values of the missing pieces $s_1,...,s_{t-1}$. To ensure that the new pieces agree with the old pieces at the missing repositories, the new polynomial's coefficients must satisfy a non-singular system of $t-1$ linear equations in t variables, thus the front-end is free to choose any value between 1 and p for K', and that choice determines all the other coefficients.

If p of the unaltered pieces are known to the adversary, however, then the threshold for the new key is effectively $t-p$ of $n-p$. If this level of security is unsatisfactory, then as long as $max(r,w,t)$ repositories are accessible, known protocols for on-the-fly reconfiguration (e.g., [11]) can be used to replace the inaccessible repositories with new, trusted repositories, using a completely new set of pieces to encrypt the key.

6. A Protocol for Preserving Integrity

So far, the protocols described here have been concerned with preserving the secrecy of data against a passive adversary. In this section we address the problem of preserving the *integrity* of data against an active adversary who can modify the local storage at fewer than t_I repositories. We wish to ensure that any such modification can be detected, implying that the compromised repository can be treated just as if it had crashed.

Our analysis is based on the following assumptions.

- The cleartext is encrypted together with a checksum that provides enough internal redundancy to detect any direct modifications to the ciphertext. Rabin and Karp have given such a checksum [21]. Another approach to this method is given by the more expensive technique of probabilistic encryption [9].
- The integrity threshold t_I is less than or equal to t (for private key SQC) or t_E (for public key SQC). Otherwise the problem is clearly hopeless.
- The adversary can, however, "spoof" the repository by taking a snapshot of its data, later replacing the current data with the out-of-date snapshot.

The basic idea is quite simple: We use either the public or private key SQC protocol, but instead of requiring quorum intersections to be non-empty, we require that quorum intersections have cardinality at least t_I. For files, this constraint ensures that each read quorum includes at least one uncompromised repository with the file's current value. Since the adversary is restricted to replaying old values with earlier timestamps, the version with the latest timestamp will be the correct one. In the counter example, each Value quorum will include at least one copy of each Inc and Dec entry.

This protocol may seem expensive, since at least one operation must have larger quorums, but the following argument (given here for files) shows that this protocol is optimal in the sense that any weaker constraint on quorum intersection yields a protocol vulnerable to spoofing. Consider a file replicated at n repositories, with read quorums of size r and write quorums of size w, where each read quorum intersects each write quorum at x repositories: $r+w-x = n$. Let R, W, and X be disjoint sets of repositories of sizes r-x, w-x, and x, where the repositories in X are controlled by an adversary.

Consider the following scenario:

1. Client A initializes the file with value *a* at some write quorum.
2. The adversary snapshots the contents of each repository in X.
3. Client B writes the value *b* at the write quorum $W \cup X$.
4. At every repository in X, the adversary restores the snapshots taken in Step 2.
5. Client C reads the obsolete value *a* from the read quorum $R \cup X$.

In this scenario, B's write quorum intersects C's read quorum only at X, thus an adversary who controls X can spoof C into reading an obsolete value. Choosing quorums at random would provide a probabilistic guarantee of integrity by increasing the likelihood that a read quorum would intersect with the most recent write quorum at some uncompromised repository, but such a guarantee may well be too weak for applications where integrity is of concern.

7. Remarks and Related Work

Brian Randell [17, 18] posed the following question: can security and fault-tolerance be integrated in a single mechanism? This question challenges the traditional rule of thumb in security work that as a system becomes more distributed, vulnerability to security attacks increases. This paper answers Randell's question affirmatively, by giving a protocol in which security proportionally increases, with relatively low cost, as the system becomes more distributed. Unlike previous approaches, our protocol attacks the issues of security and fault-tolerance simultaneously with a single mechanism.

Tompa and Woll [23] have given a stronger version of the secret sharing protocol that is also applicable to our model. Chor *et al.* [4] have given a "verifiable secret sharing" protocol which ensures that the dealer cannot cheat. Chor and Rabin [5] have shown how this protocol can be used to generate bits with a high degree of independence in a Byzantine distributed system. The verifiable secret sharing protocol can also be incorporated into our techniques, but it would be more expensive to do so. The ITOSS [16] operating system used the secret sharing protocol to provide a distributed command processor for high security applications.

References

[1] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr.
RSA and Rabin functions: certain parts are as hard as the whole.
In *Proceedings, 25th IEEE Symposium on the Foundations of Computer Science*. November, 1984.
To appear in SIAM J. on Computing.

[2] A. Birrell.
Secure communication using remote procedure calls.
ACM Transactions on Computer Systems 3(1):1-14, February, 1985.

[3] M. Blum and S. Micali.
How to generate cryptographically strong sequences of pseudo-random bits.
SIAM J. on Computing 13(4):850-864, 1984.

[4] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch.
Verifiable secret sharing and achieving simultaneity in the presence of faults.
In *Proceedings, 26th IEEE Symposium on the Foundations of Computer Science*. October, 1985.

[5] B. Chor and M. Rabin.
Achieving independence in a logarithmic number of rounds.
1987.
To appear, PODC 87.

[6] W. Diffie and M. E. Hellman.
New Directions in Cryptography.
IEEE Transactions on Information Theory IT-22(6):644-654, Nov., 1976.

[7] D. K. Gifford.
Weighted Voting for Replicated Data.
In *Proceedings of the Seventh Symposium on Operating Systems Principles.* ACM SIGOPS, December, 1979.

[8] O. Goldreich, S. Micali, and A. Wigderson.
Proofs that yield nothing but the validity of their assertion.
Preprint.

[9] S. Goldwasser and S. Micali.
Probabilistic encryption and how to play mental poker.
In *Proceedings of the 14th Symposium on the Theory of Computation*. May, 1982.

[10] S. Goldwasser, S. Micali, and C. Rackoff.
The knowledge complexity of interactive proofs.
In *Proceedings of the 17th Symposium on the Theory of Computation*. May, 1985.

[11] M. P. Herlihy.
A quorum-consensus replication method for abstract data types.
ACM Transactions on Computer Systems 4(1), February, 1986.

[12] L. Lamport.
Time, clocks, and the ordering of events in a distributed system.
Communications of the ACM 21(7):558-565, July, 1978.

[13] C. Meyer and S. Matyas.
Cryptography.
Wiley, 1982.

[14] R. Needham and M. Schroeder.
Using encryption for authentication in large networks of computers.
Communications of the ACM 21(12), December, 1978.

[15] M. Rabin.
Digitalized signatures and public-key functions as intractable as factorization.
Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, MIT, January, 1979.

[16] M. Rabin and J. D. Tygar.
An integrated toolkit for operating system security.
Technical Report TR-01-87, Aiken Computation Laboratory, Harvard University, January, 1987.

[17] B. Randell.
Recursively structured distributed computing systems.
In *Proceedings, Third Symposium on Reliability in Distributed Software and Database Systems*, pages 113-118. October, 1983.

[18] B. Randell and J. Dobson.
Reliability and security issues in distributed computing systems.
In *Proceedings, Fifth Symposium on Reliability in Distributed Software and Database Systems*, pages 113-118. January, 1986.

[19] J. Reif and J. D. Tygar.
Efficient parallel pseudo-random number generation.
In *Advances in Cryptology: CRYPTO-85*, pages 433-446. Springer-Verlag, August, 1985.
To appear in SIAM J. on Computing.

[20] R. Rivest, A. Shamir, and L. Adleman.
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.
Communications of the ACM 21(2):120-126, Feb., 1978.

[21] Richard M. Karp and Michael O. Rabin.
Efficient Randomized Pattern-Matching Algorithms.
Technical Report TR 31-81, Aiken Laboratory, Harvard University, December, 1981.

[22] A. Shamir.
How to share a secret.
Communications of the ACM 22(11):612-614, Nov., 1979.

[23] M. Tompa and H. Woll.
How to share a secret with a cheater.
In *Advances in Cryptology: CRYPTO-86.* Springer-Verlag, 1986.
Also available as IBM Research Report RC-11840.

A Study of Password Security

Michael Luby †
Charles Rackoff ‡
University of Toronto

1. Introduction

Our work is motivated by the question of whether or not the password scheme used in UNIX is secure. The following password scheme is a somewhat simplified version of the actual password scheme used in UNIX. We feel that this simplified version captures the essential features of the actual password scheme used in UNIX. When a user logs in for the first time he creates a random password and types his user name together with the password into the system. The system creates an encryption of the password using the Data Encryption Standard (DES) and stores this (only the encryption, not the password) together with the user name in a password file. Thereafter, whenever the user logs in and types in his user name and password the system computes the encryption of the password and only allows the user to successfully log in if the encryption matches the entry stored with the user name in the password file.

The system model guarantees that the password file is write protected but not necessarily read protected. We do not formally justify this model of the system, but only remark that perhaps the reasoning behind this model is that an unauthorized user may be able to read the password file when the system is unprotected (e.g. during a crash) but the system is able to keep enough backup copies of the password file so that even if an unauthorized user can write to one copy of the file, he will not be able to update all copies. Informally, the password scheme would be secure if any unauthorized user, who has a copy of the password file, cannot generate a password whose encryption is stored in the password file.

The following is a more complete description of the password scheme discussed above. The password is a bit string x of length 56 and the encryption of x is a bit string y of length 64, where y is DES evaluated on 0^{64} (the bit string consisting of 64 zeroes) using key x. It is stated informally in [De] that this password scheme is secure if DES is secure when used in a private key cryptosystem. We formally investigate this question by the following approach.

† research partially supported by the Canadian Natural Sciences and Engineering Research Council operating grant A8092 and by a University of Toronto research grant

‡ research partially supported by the Canadian Natural Sciences and Engineering Research Council operating grant A3611

Since we cannot even prove that DES is secure in any formal sense when used in a block private key cryptosystem, we study the security of a UNIX-like password scheme when in place of DES we use a pseudo-random function generator (see [GGM1], [GGM2], [LR1], [LR2] for definitions, existence of, and applications of pseudo-random function generators and pseudo-random permutation generators).

Let n be the length of the encryption of a password and let $l(n)$ be the length of the password. Very informally stated, what we prove is that if a pseudo-random function generator with key length $l(n) \leq n + O(\log n)$ is used in a UNIX-like password scheme then the password scheme is secure. On the other hand, we describe a pseudo-random function generator with key length $l(n) \geq n + h(n)$ where $\log n = o(h(n))$, such that the UNIX-like password scheme it generates is not at all secure. The implication of this latter result is that if the password is too much longer than the encryption of the password, then even if we have a cryptosystem which is secure when used as a block private key crytosystem, the UNIX-like password scheme it generates may not be at all secure. As a more concrete example, we show that a modified version of DES, which most people believe is even more secure than DES when used in a block private key cryptosystem and also more pseudo-random, is not at all secure when used in the UNIX-like password scheme. This is a lesson against the blind philosophy of taking something which is proven secure in one sense and using it where it must be secure in a different sense and assuming without actually proving that it is secure in the different sense.

2. Definition of Secure Password Scheme

Let *poly* be the set of all polynomial functions.

Password Scheme Definition: g is a password scheme with password length function $l \in poly$, where $g = \{g^n : n \in N\}$, where for each $n \in N$, g^n is a function from $l(n)$ bits to n bits.

Intuition: For each $n \in N$, the password is of length $l(n)$ and the encryption of the password is of length n.

Security Definition: A password scheme g is secure if there is no polynomial size family of circuits which can for infinitely many values of $n \in N$, receive an encryption of a randomly chosen password and output a password which has the same encryption with probability greater than $\frac{1}{n^c}$ for some constant $c > 0$. More precisely, g is secure if there is no polynomial size family of circuits (where the n^{th} circuit in the family is to be used to evaluate inputs of length n) A which has the following properties. On an input of length n, A produces an output of length $l(n)$. We say that A is **successful** on input y if A produces a string z such that $g^n(z) = y$. For infinitely many $n \in N$, A has the property that when x is a bit string of length $l(n)$ chosen randomly the probability that A is successful on input $g^n(x)$ is at least $\frac{1}{n^c}$.

Comment: There is another definition of security where "probabilistic polynomial time algorithm" is substituted for "polynomial size family of circuits" in the definition. Our theorem is true (using a similar proof) with respect to this definition when "probabilistic polynomial time algorithm" is substituted for "polynomial size family of circuits" in the definition of pseudo random function generator.

Comment: Informally, we want the security of g to reflect the fact that no polynomial size family of circuits given the encryptions of a polynomial number of randomly chosen passwords can produce even one password which has the same encryption as one of the given encryptions. It can be easily shown that if g is secure in the sense defined above then g is secure in the alternative sense defined below, which reflects this informal notion of security.

Alternative Security Definition: Exactly the same as for the security definition except the properties of the circuit family A are modified as described here. Let $b \in poly$ and let A be a polynomial size family of circuits similar to A described above except that it has $b(n)$ inputs of length n and one output of length $l(n)$. A is successful on a particular input if it successfully produces a string z of length $l(n)$ such that $g^n(z)$ is equal to one of the $b(n)$ input strings.

UNIX-like Password Scheme: Let f be a function generator with key length $l \in poly$ (see [GGM1], [GGM2], [LR1] or [LR2] for a definition of a function generator and the definition of a pseudo-random function generator). For each $n \in N$, the encryption of a password x of length $l(n)$ is $f_x^n(0^n)$.

2. The Main Theorem

Theorem: If f is a pseudo-random function generator and $l(n) \leq n + d\log n$ for some constant d and for all sufficiently large $n \in N$, then the UNIX-like Password Scheme is secure.

Proof: We assume for contradiction that the password scheme is not secure. Thus, for some constant $c > 0$ there is a polynomial size family of circuits A which for some infinite $N_1 \subseteq N$ has the following characteristics. For each $n \in N$, let $\varepsilon(n)$ be the probability that A is successful when a password x of length $l(n)$ is chosen at random and the input to A is $f_x^n(0^n)$. For each $n \in N_1$, $\varepsilon(n) \geq \frac{1}{n^c}$. We show how to construct a polynomial size family of circuits C, one circuit for each $n \in N_1$, which demonstrates that f is not pseudo random.

For each $n \in N$, define $\varepsilon'(n)$ to be the probability that A is successful when a string y of length n is chosen at random and the input to A is y. For each $n \in N_1$, we consider two cases:

Case 1: $\varepsilon'(n) \le \frac{\varepsilon(n)}{2}$. In this case $\varepsilon(n) - \varepsilon'(n) \ge \frac{1}{2n^c}$. Let F^n be the set of all functions from n bits to n bits. We define the circuit with index n in the circuit family C in terms of A, which is to distinguish F^n from f^n as follows:

```
**Circuit C:** The input to $C$ is a function $f_1$.
$C$ computes $\alpha = f_1(0^n)$.
$C$ computes $x = A(\alpha)$.
$C$ computes $y = f_x^n(0^n)$.
If $y = \alpha$ then $C$ outputs 1, otherwise $C$ outputs 0.
```

If f_1 is randomly chosen from f^n, then the probability that C outputs 1 is $\varepsilon(n)$. If f_1 is randomly chosen from F^n, then the probability that C outputs 1 is $\varepsilon'(n)$. Thus, C distinguishes between F^n and f^n with probability at least $\varepsilon(n) - \varepsilon'(n) \ge \frac{1}{2n^c}$.

Case 2: $\varepsilon'(n) \ge \frac{\varepsilon(n)}{2}$. In this case $\varepsilon'(n) \ge \frac{1}{2n^c}$. We define the circuit with index n in the circuit family C in terms of A, which is to distinguish F^n from f^n as follows:

```
**Circuit C:** The input to $C$ is a function $f_1$.
$C$ computes $\alpha = f_1(0^n)$.
$C$ computes $x = A(\alpha)$.
$C$ computes $y = f_x^n(1^n)$.
$C$ computes $\beta = f_1(1^n)$.
If $y = \beta$ then $C$ outputs 1, otherwise $C$ outputs 0.
```

If f_1 is randomly chosen from f^n, then by the claim below the probability that C outputs 1 is at least $\frac{1}{2n^{c+d}}$. If f_1 is randomly chosen from F^n, then the probability that C outputs 1 is $\frac{1}{2^n}$. Thus, C distinguishes between F^n and f^n with probability at least $\frac{1}{2n^{c+d}} - \frac{1}{2^n} \ge \frac{1}{4n^{c+d}}$ for sufficiently large n.

Claim: When x is a randomly chosen string of length $l(n)$ then the probability that $A(f_x^n(0^n)) = x$ is at least $\frac{1}{2n^{c+d}}$.

Proof of Claim: Let M be the set of strings x of length $l(n)$ such that A is successful on input $f_x^n(0^n)$. Let M' be the set of string y of length n such that A is successful on input y. Let S be the set of strings x of length $l(n)$ such that $A(f_x^n(0^n)) = x$. Since A is a one to one onto function from M' to S, $|S| = |M'|$. It is easy to see that $\frac{|M|}{2^{l(n)}} = \varepsilon(n)$ and $\frac{|M'|}{2^n} = \varepsilon'(n)$. Thus, since $l(n) \le n + d\log n$,

$$\frac{|S|}{2^{l(n)}} = \frac{|M'|}{2^{l(n)}} \ge \frac{\varepsilon'(n)}{2^{d\log n}} \ge \frac{1}{2n^{c+d}}.$$

□

From case 1 and case 2 it is easy to see that for almost all $n \in N_1$, the circuit family C distinguishes f^n from F^n with probability at least $\frac{1}{4n^{c+d}}$. Thus, the function generator f is not pseudo-random. □

Comment: Levin [Le] proves a similar theorem to this, in a completely different context, in the case when $l(n) \le \frac{n}{2}$.

3. The Password Should not be Too Long

In this section, we give examples which show that the theorem proved in the previous section is the strongest general theorem possible with respect to the length of the password and the encryption of the password. Our first example shows why a pseudo-random function generator with a long key length cannot necessarily be used to produce a secure UNIX-like password scheme. The second example, which is a very practical example, demonstrates that a cryptosystem which people believe to be even more secure than the Data Encryption Standard, when used in a UNIX-like password scheme, produces a totally insecure password scheme.

The first example is as follows. Let f be a pseudo-random function generator with key length $k(n)=n$. We define a function generator g in terms of f as follows. The key length function for g is $l(n)=n+\log^2 n$ ($\log^2 n$ was chosen to be such that for all $h \in poly$, for sufficiently large n, $\frac{1}{2^{\log^2 n}} \le \frac{1}{h(n)}$). For each $n \in N$, let x be a string of length $l(n)$ which is the concatenation of a string x_1 of length $\log^2 n$ and a string x_2 of length n. For all strings $a \ne 0^n$ of length n, $g_x^n(a)=f_{x_2}^n(a)$. $g_x^n(0^n)$ is defined to be $f_{x_2}^n(0^n)$ if $x_1 \ne 0^{\log n^2}$ and is defined to be x_2 if $x_1 = 0^{\log^2 n}$. It is not hard to prove that g is pseudo-random. On the other hand, if g is used in a UNIX-like password scheme then the resulting password scheme is totally insecure. To see this, note that for any encrypted password y of length n, the password consisting of $0^{\log^2 n}$ concatenated with y always encrypts to y.

The second very practical example is the following. The key for DES is 56 bits long. This key is expanded in a predetermined way into 16 keys each of length 48, and the 16 keys are used in the 16 levels of encryption in DES. Let MDES, (mnemonic for Modified Data Encryption Standard) be the same as the Data Encryption Standard except that it uses 16 independent keys of length 48, and these 16 keys are used in the 16 levels of DES. Most people believe that MDES is at least as secure as DES when used in a block private key cryptosystem. Most people believe that MDES is as pseudo-random as DES. On the other hand, it is easy to see that if MDES is used in a UNIX-like password scheme then the resulting password scheme is totally insecure. To see this, suppose that we want to find a password which has the same encryption as a particular string y of length 64. Let y_1 be the first half of y and let y_2 be the second half of y. The password is chosen by first arbitrarily choosing the first 14 keys of length 48. The 15^{th} key is computed such that the left hand side of the final encryption matches y_1. Because of the way each level of DES is computed, the 15^{th} key can be easily computed for any set of values of the first 14 keys and for any value of y_1. Similarly, the 16^{th} key is computed such that the right hand side of the final encryption matches y_2. For exactly the same reasons, the 16^{th} key can be easily computed for any set of values of the first 15 keys and for any value of y_2.

4. Acknowledgements

We thank Paul Beame, Johan Hastad and Gilles Brassard for some helpful discussions and simplifications to this work.

5. References

[De] Denning, D., *Cryptography and Data Security*, January 1983, Addison-Wesley Publishing Company, Inc.

[GGM1] Goldreich, O., Goldwasser, S., Micali, S., *How to Construct Random Functions*, Proceedings of the 25th *Annual Symposium on Foundations of Computer Science, October 24-26, 1984*

[GGM2] Goldreich, O., Goldwasser, S., Micali, S., *How to Construct Random Functions*, J. for Association of Computing Machinery, Vol. 33, No. 4, October 1986, pp. 792-807 of Computer Science, October 24-26, 1984

[Le] Levin, L.A., *One-Way Functions and Pseudorandom Generators*, Proceedings of the 17th ACM Annual Symposium on Theory of Computing, May 6-8 1985, pp. 363-365.

[LR1] Luby, M., Rackoff, C., *Pseudo-random Permutation Generators and Cryptographic Composition*, Proceedings of the 18th *ACM Annual Symposium on Theory of Computing, May 28-30, 1986*

[LR2] Luby, M., Rackoff, C., *How to Construct Pseudo-random Permutations from Pseudo-random Bits*, to appear in special issue on Cryptography, SIAM J. on Computing

A Video Scrambling Technique Based On Space Filling Curves

(Extended Abstract)

Yossi Matias and Adi Shamir

Department of Applied Mathematics, The Weizmann Institute of Science
Rehovot 76100, *Israel*

Abstract

In this paper we propose a video scrambling technique which scans a picture stored in a frame buffer along a pseudo-random space filling curve. We describe several efficient methods for generating cryptographically strong curves, and show that they actually decrease the bandwidth required to transmit the picture.

1. Introduction.

Video signals play a predominant role in modern society in diverse applications such as entertainment, telecommunications and military surveillance. In many applications it is essential to guarantee the privacy of this signal. However, the standard cryptographic techniques may be inadequate for three basic reasons :

1) The transmitted signal is analog.
2) The transmission rate is very high.
3) The allowable bandwidth is limited, and can be accommodated only when the grey levels of adjacent pixels are highly correlated.

Wyner [W1,W2] suggested a method of scrambling a discrete time analog sequence by using a large family of linear orthogonal invertible transformations which he described, that result in a negligible expansion of bandwidth. However, these transformations are not easy to compute, and in the case of video signals they do not exploit the two dimensional nature of the signal.

A video picture is usually transmitted by scanning its elements (pixels). The frame is scanned from top to bottom and from left to right; this is the conventional raster scan. Our approach is to effect a pixel permutation by changing the scanning order without changing the pixels' values. The scanning order is determined by the encryption key, and should be changed occasionally (but not necessarily every frame). Since we want to preserve the correlation between the grey levels of adjacent transmitted pixels, we propose to scan the picture with a connected curve i.e. a Space Filling Curve (SFC).

The encryption method is as follows :

a. Pseudo-randomly choose a Space Filling Curve.

b. Transmit an analog signal that represents the grey levels of successive pixels along this curve.

The corresponding decryption method is :

a. Choose the same SFC (by using a shared cryptographic key).

b. Fill a frame buffer with the received signal by following the SFC.

We can use the interframe redundancy and generalize our scheme to 3 dimensions. This is done by storing several consecutive frames in several frame buffers and finding a random SFC for the 3 dimensional collection of frame buffers. The use of a 3-dimensional SFC increases the number of possible SFCs by many orders of magnitude, increases the confusion of a would be attacker and further improves the signal's bandwidth, but increases the cost and complexity of the implementation.

Let G be the infinite graph whose vertex set consists of all points of the plane with integer coordinates and in which two vertices are connected if and only if the (Euclidean) distance between them is equal to 1. A *grid* graph is a vertex induced subgraph of G. Let v_x and v_y be the *coordinates* of the vertex v. A *rectangular grid* graph $R(m,n)$ is a grid graph whose vertex set is $\{v : 1 \leq v_x \leq m \text{ and } 1 \leq v_y \leq n\}$.

We shall consider the frame-buffer as a rectangular grid graph $R(m,n)$, whose vertices are the pixels. It follows that the Space Filling Curve is a Hamiltonian path in $R(n,m)$. The problem of finding Hamiltonian paths between specified pairs of vertices in rectangular grid graphs was examined by Itai et al [IPS], who showed how to construct one path. Our goal is to find many Hamiltonian paths, and we do not require specific endpoints. The ideal, of course, is to have an efficient algorithm which produces all Hamiltonian paths. However, a large family of Hamiltonian paths may be sufficient if we are careful about its cryptographic quality. To see what can go wrong, consider the following algorithm for generating Hamiltonian paths:

Algorithm A.

- Start at the upper left corner of the rectangle and move to the right.

  Repeat until all vertices are visited :

- Move straight ahead until visiting all unvisited vertices in this direction. At the last vertex decide (by the next pseudo-random bit) : either turn a 90° turn, or make a U-turn (towards unvisited vertices).

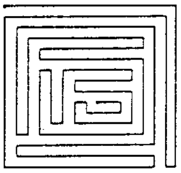


Figure 1. An example of a SFC generated by Algorithm A in a 16 by 16 rectangular grid.

Algorithm A is very efficient, it generates a large family of $\binom{n+m-2}{n-1}$ Hamiltonian paths, and it can be easily implemented. A typical scanning order produced by this algorithm is given in *Fig.* 1. However, this cryptosystem is easy to break with a ciphertext-only attack. Notice that we have long lines and turning points of two kinds: one is the 90° turn, and the other is the U-turn (180 degrees turn). Adjacent vertices from two lines which are connected by a U-turn are highly correlated. In order to locate the points where U-turns have been taken, it is sufficient to test for local symmetry points along the transmitted path **i.e.** for segments whose values are almost palindromes. Finding these points is quite easy, and thus breaking the cryptosystem is straightforward.

We conclude the introduction with a remark about the number of SFCs U_N in a square lattice of N points. By Orland et al. [OID] we have (for large N)

$$U_N \sim \begin{cases} 1.4715^N = 2^{0.5573N} & \text{in 2D square lattice} \\ 2.2073^N = 2^{1.1423N} & \text{in 3D cubic lattice.} \end{cases}$$

This demonstrates the advantage of using a 3-dimensional SFC. The confusion of a naive attacker is enlarged by a factor of

$$\frac{U_N^{(3D)}}{U_N^{(2D)}} = 1.5^N = 2^{0.585N} \quad (!!!)$$

which is more than 2^{150000} for $N = 512 \times 512$.

2. Tile and Merge

The method of producing Hamiltonian circuits discussed in this section is based on the concept of first *covering* all vertices by disjoint elementary circuits, and then *merging* these circuits into a single Hamiltonian circuit. Since our aim is to produce many Hamiltonian circuits we shall introduce randomness both in the *covering* and in the *merging* processes. While it is easy to produce all the possible *mergings* of a certain *cover* of a grid graph using the concept of spanning trees, the production of every possible *cover* (with a probability greater then zero) seems to be hard.

By using the tile-and-merge method we introduce efficient algorithms for producing exponentially many Hamiltonian circuits in a rectangular grid graph. A large subset of these Hamiltonian circuits can be encoded by only 1 bit per pixel. The algorithms can be easily extended to 3D.

2.1 The General Method

Let G be an undirected graph. Consider disjoint circuits $C_1, C_2, \ldots, C_k$ that cover all the vertices of G. Define two circuits C_i and C_j to be *neighbors* if there is an edge $e_1 = (v_1, v_1')$ in C_i , and an edge $e_2 = (v_2, v_2')$ $(= (v_2', v_2)$) in C_j such that there are edges $f_1 = (v_1, v_2)$ and $f_2 = (v_1', v_2')$ in G. Note that f_1 and f_2 complete e_1 and e_2 into a square. If C_i and C_j are neighbors then we can connect them into one circuit C_{ij} which covers exactly the same vertices as C_i and C_j by replacing e_1 and e_2 by f_1 and f_2. Denote this replacement as a *legal replacement* and denote (e_1, e_2) as a *legally replaced couple (LRC)* in (C_i, C_j). Note that between 2 circuits there may be several legal replacements but only one of them may be (randomly) chosen. Thus by a single legal replacement we reduce the number of circuits by 1. If we start with k circuits then after $k-1$ legal replacements we are left with a Hamiltonian circuit. This leads directly to the following definitions and algorithm for finding a random Hamiltonian circuit in a general grid graph.

Let G be a graph. Given k disjoint circuits $C_1, C_2, \ldots, C_k$ over G, define a graph $G_k = (V_k, E_k)$ where $V_k = \{C_1, C_2, \ldots, C_k\}$ and $E_k = \{(C_i, C_j) : C_i \text{ and } C_j \text{ are neighbors }\}$. For each $(C_i, C_j) \in E_k$ let L^k_{ij} be: $L^k_{ij} = \{(e_1, e_2) : e_1 \text{ and } e_2 \text{ are LRC in } (C_i, C_j)\}$. Note that in a grid graph, edges of a legally replaced couple are adjacent edges that are located in different circuits. Denote two circuits to be *external* if neither of them is surrounded by the other. The circuits $C_1, C_2, \ldots, C_k$ are called a *tile* of G. If the circuits are all external in pairs then we say that they are *external circuits* and we denote the tile as an *external tile.*

Algorithm SPAN

Given a grid graph G which has a Hamiltonian circuit:

1) Choose k disjoint external circuits $C_1, C_2, \ldots, C_k$ that **Cover** all the vertices of G s.t. G_k (as defined above) is connected.

2) Randomly choose a spanning tree for G_k.

3) For each edge (C_i, C_j) in the spanning tree randomly choose a legally replaced couple in L^k_{ij} and perform the legal replacement.

◇ ◇ ◇

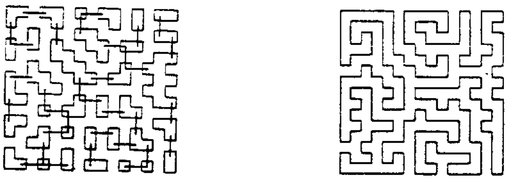


Figure 2. (a) An example of covering circuits and a spanning tree over them in a 16 *by* 16 *rectangular grid. (b) The resulting Hamiltonian circuit.*

THEOREM 1. The output circuit of algorithm **Span** is a Hamiltonian circuit in G.

Note that the fact that the initial tile is external is necessary for the correctness of the algorithm, since G_k may become disconnected during the execution of the algorithm when some of the circuits are internal.

Algorithm **Span** can be generalized so that it can find more Hamiltonian circuits after a *cover* was selected i.e. to enhance the randomness of the output circuit. This may be done by a slight modification of the algorithm with an insignificant degradation in efficiency. The main idea is to use an adaptive process that takes into account the new edges (in G_k) and LRCs which are created after each replacement of edges (in G).

If we could randomly choose all the combinations of elementary disjoint circuits that cover all the vertices in G then we would have a complete solution to the problem of finding all the Hamiltonian circuits, but this seems to be a hard problem. On the other hand there are some choices we can easily make. The most trivial choice is to make all C_i squares of 4 vertices (for the sake of simplicity we assume that the dimensions of the grid are even). The next natural choice is to make all C_i rectangles of 6 vertices. Note that in this case a 'legal replacement' as done in step 3 of **Span** can often be done in two ways; one of them should be randomly selected. A combination of the two choices results in an exponential number of tilings ($2^{\frac{mn}{8}}$).

2.2 Tile by Squares*

When G is tiled-by-squares only (the trivial choice), **Span** generates a very small subset of the set of all Hamiltonian circuits. Still, using the tile-by-square procedure **Span** constructs a set of Hamiltonian circuits with exponential cardinality. Moreover, the tile-by-squares procedure has a large advantage in coding the circuit. In addition, it can be conveniently analyzed and is the natural basis for extensions to more general graphs.

A simple and convenient method of encoding the scanning order is to add to each pixel the information of the next step in the scan. Thus, after arriving at a pixel, we can use this information and a Finite State Machine in order to know what is the next pixel in the scan. Obviously for any SFC we can locally use $\log_2 3 \approx 1.585$ bits per pixel (on the average) to represent *Forward, Left*

* In the beginning of our research Ron Rivest had communicated to us an idea of generating a Hamiltonian circuit in a $2n$ by $2n$ rectangular grid graph G in two steps:

1) Finding a spanning tree for the n by n rectangular grid graph, for which each vertex is the center of a square in G.
2) Surrounding the spanning tree using the edges of G.

Algorithm **Span** restricted to the tile-by-squares strategy produces exactly the same Hamiltonian circuits.

or *Right*. However, if we could save the information with only 1 bit per pixel then this could be significant in practical implementations.

CLAIM 1. The SFCs that are generated by algorithm **Span** using tile-by-squares can be (constructively) encoded by only 1 bit per pixel.

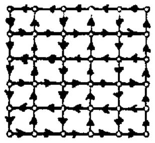


Figure 3. A 6 by 6 Manhattan orientation rectangular grid graph

Let G^M be a directed grid graph with a Manhattan orientation as in *Fig.* 3.

CLAIM 2. The set of Hamiltonian circuits that can be generated by algorithm **Span** restricted to tile-by-squares is exactly the set of Hamiltonian circuits in G^M.

Using Kasteleyn's exact enumeration [Kas] we have

COROLLARY 1. The number of Hamiltonian circuits that can be constructed by **Span** using tile-by-squares is $2^{0.4206334N}$.

Considering the cryptographic strength of the scheme we have

COROLLARY 2. Given a picture of black&white (horizontal or vertical) strips of one pixel width scrambled using a SFC, we can easily reconstruct the SFC that was used.

The SFC which is constructed by a tile-by-squares can be exposed by a single black&white pattern which is simple and therefore has a reasonable probability to occur (at least locally) in real transmissions. Thus, although the tile-by-squares technique, when used in algorithm **Span**, is economically attractive, it provides a much weaker scheme than the more general techniques which tile the plane with elementary circuits of various sizes and shapes.

3. The Ham+ Algorithm

In this section we describe an efficient algorithm which can produce all the Hamiltonian circuits of the rectangular grid $R(m, n)$ plus 'almost Hamiltonian' circuits that miss up to $\frac{n}{3}$ vertices in the $(m-1)$'th row. Every Hamiltonian circuit has a positive probability to be generated in each run, and in our application the almost Hamiltonian circuits are just as useful as the Hamiltonian circuits.

3.1 General Description

Algorithm HAM+ constructs the Hamiltonian circuit by scanning the rows of the array from top to bottom. We shall enumerate the rows from top to bottom and the columns from left to right. After the $(i-1)$'th iteration the rectangle $R(i-1, n)$ is completely covered with disjoint paths whose endpoints are all in the i'th row. In the i'th iteration the i'th row is filled by advancing the endpoints. No endpoint is left inside $R(i, n)$ and no path becomes a circuit. This is accomplished by letting the endpoints of each path identify each other as *partners*. In the final two rows we tie the paths together in order to get a global circuit, but in the process we may miss some of the vertices in the $(m-1)$'th row. We do not know how to complete the process without this additional degree of freedom.

When stationed at a vertex, we choose the edges connected to it in the circuit by choosing two of its neighbors. The idea is to pseudo-randomly choose between all possible choices in a way that would not prevent us from successfully generating the promised circuit. However, it is not guaranteed that all circuits have the same probability to be generated because when stationed at a vertex in the i'th row, the probability of an edge to be chosen should be dependent on the $(i-1)$ rows; this fact is not taken into consideration in the algorithm.

A full description of algorithm Ham+ (along with its non-obvious proof of correctness) will be given in the full paper.

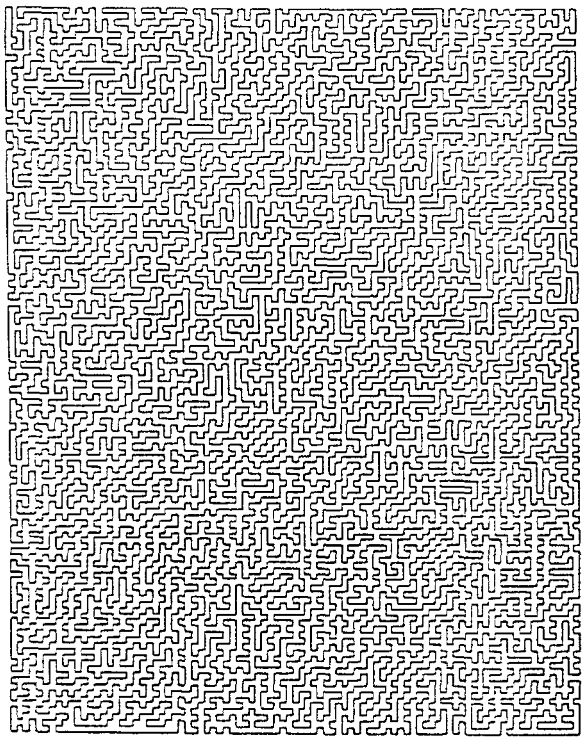


Figure 4. A circuit that was generated by Ham+ *(note the 6 missing vertices at the last but one row).*

3.2 Discussion

The number of missed vertices is $N_{missed} \leq \lfloor \frac{n}{3} \rfloor$. It is possible to reduce the number of missed vertices if we begin to tie the disjoint paths in the $(m-2)$'th or earlier rows. An interesting question that is left open is what is the minimal lookahead required to generate exactly the set of all Hamiltonian circuits?

In the application of HAM+ for video scrambling, the problem of missed vertices can be overcome by several techniques. For instance by transmitting the rectangle $R(m-2,n)$. The relative number of missed vertices $\frac{N_{missed}}{nm}$ is less then $\frac{1}{3m}$ (which is 0.065% for $m = 512$). Therefore the influence of the missed vertices on the overall performance is negligible.

The complexity of the algorithm is $O(mn)$, since each pixel is considered only once. The algorithm is efficient, it can be easily implemented and it can generate all Hamiltonian circuits plus extra circuits that can be used as well.

4. Autocorrelations of a Random 'Typical' Picture's Signals

We wish to examine the autocorrelation function $\varrho(k)$ of the scrambled signal and compare it with the autocorrelation $\rho(R)$ of the original signal. Let $p_k(R)$ be the probability that two points in the SFC that are k seperated have the Euclidean distance R in the grid and let E_k be the expectation with respect to $p_k(R)$. For a random picture of an isotropic stationary model (where the two-dimensional autocorrelation function depends only on the Euclidean distance between points) we have

$$\varrho(k) = E_k(\rho(R)) \tag{0.1}$$

for $k \ll \{$ the number of pixels in a line $\}$.

It was shown in [Ren] that

$$p_k(R) = A_k \left(\frac{R}{k^\nu}\right)^\theta \exp\{-\left(\frac{R}{k^\nu}\right)^\delta\} \tag{0.2}$$

where $\nu = \frac{3}{4}, \theta = 1.93\pm0.27, \delta = 4.6\pm0.06$ in two dimensions and $\nu = 0.59, \theta = 0.67\pm0.34, \delta = 2.6\pm 0.06$ in three dimensions; A_k is a normalizing factor. Recall the form of the autocorrelation function of a (simplified) isotropic model (a first-order Markov process) [NL] $\rho(R) = e^{-\alpha R}$. Together with Eq. (0.1) and (0.2) it implies

$$\varrho(k) = A_k \sum_R \left(\frac{R}{k^\nu}\right)^\theta \exp\{-\left(\frac{R}{k^\nu}\right)^\delta - \alpha R\}. \tag{0.3}$$

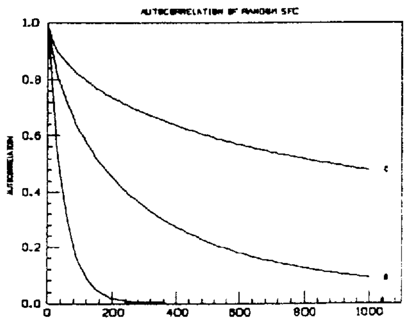


Figure 5.

In *Fig.* 5 we show the autocorrelation functions of a raster scan signal (A), a random two-dimensional SFC scan signal (B) and a random three-dimensional SFC scan signal (C). The higher autocorrelations in the SFC scan signals make it possible to reduce the bandwidth required to transmit the encrypted signal compared to the bandwidth required to transmit the original signal (or alternatively, for a given bandwidth, to gain a picture of better quality when its signal is scrambled before transmission). Unlike most other analog encryption schemes, our scheme actually compresses the signal. A related compression algorithm (which cannot be used as a cryptosystem) was recently proposed in [LZ].

5. Demonstrative Simulations

Faithful to the saying "seeing is believing" we applied our scheme to two pictures that are considered typical examples: A detailed *Landscape* picture and a *Head & Shoulder* picture. In order to simulate the effect of a low pass filter of $p\%$ on a transmitted video signal we applied a Fourier Transform on the (one dimensional) signal, zeroed all $(100 - p)\%$ high frequency coefficients, and performed an inverse Fourier Transform of the result.

Each picture was transformed into a signal using four scans: a (conventional) Raster Scan, two scans along different SFCs generated by **Ham+**, and a Random Scan in which the pixels in each row were randomly permuted. Each signal was filtered with low pass filters of $p\%$ with $p = 80, 60, 40, 20, 10, 5, 1$.

In *Fig. 8.a* we can see how the video signals of the Landscape picture, obtained by the four scans, are displayed on a screen when a descrambler is not used. The plain signal appears at the upper row of the figure. The signals that were obtained by the SFCs scans appear at the second and third rows of the figure. The signal that was obtained by the Random scan appears at the lower row of the figure.

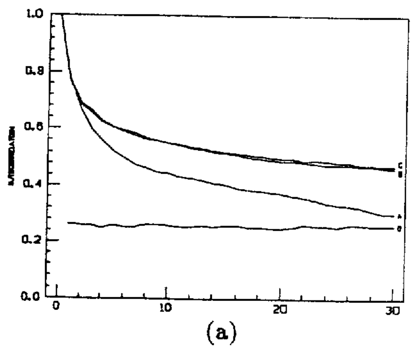
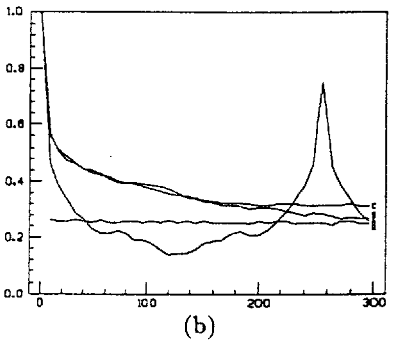



Figure 6. The autocorrelations of the Raster Scan signal (A), the SFCs' signals (B and C) and of the Random Scan signal in the Landscape picture in the range 0–30 (a) and 0–300 (b).

Fig. 6 shows the behavior of the autocorrelation functions in the ranges 0–30 and 0–300. The Raster Scan's function behaves as expected – it rises to a second peak at 256 due to the line-to-line correlation; refer to Franks [Fr]. The SFCs' functions continue the descent, demonstrating the loss of the line-to-line correlations; this loss occurs while on the other hand, there is an enhancement of correlation in the range 0-230 (in comparison with the Raster Scan's function). The Random Scan's function looks like a δ-function.

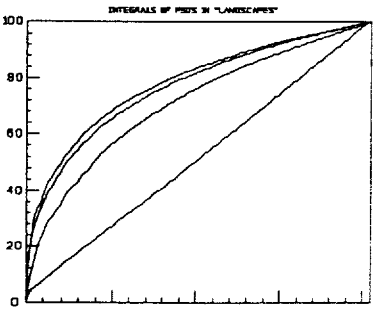


Figure 7. The integrals of the PSDs of the 4 signals.

The bandwidths of the 4 scans are demonstrated by showing the integrals I_ps of the PSDs. An I_p shows the amount of power that remains in a signal after a low pass filtering is applied; therefore, the higher the I_p, the better is the scan. *Fig.* 7 shows all 4 I_ps. The integral of the Random Scan's PSD is, as expected, a straight line. The Raster Scan's PSD is much better; it is the curve just above the Random Scan's line. However, it is not as good as the PSDs of the SFCs which are the two top curves.

The difference between the SFCs' I_ps and the Raster Scan's I_p is especially emphasized in the range of about 5–25% of the spectrum. This is the range where we expect the most significant improvements of performance of the scrambled signal in comparison with the original signal; i.e. we get a picture of better quality if we scramble it before transmission via a low-pass channel.

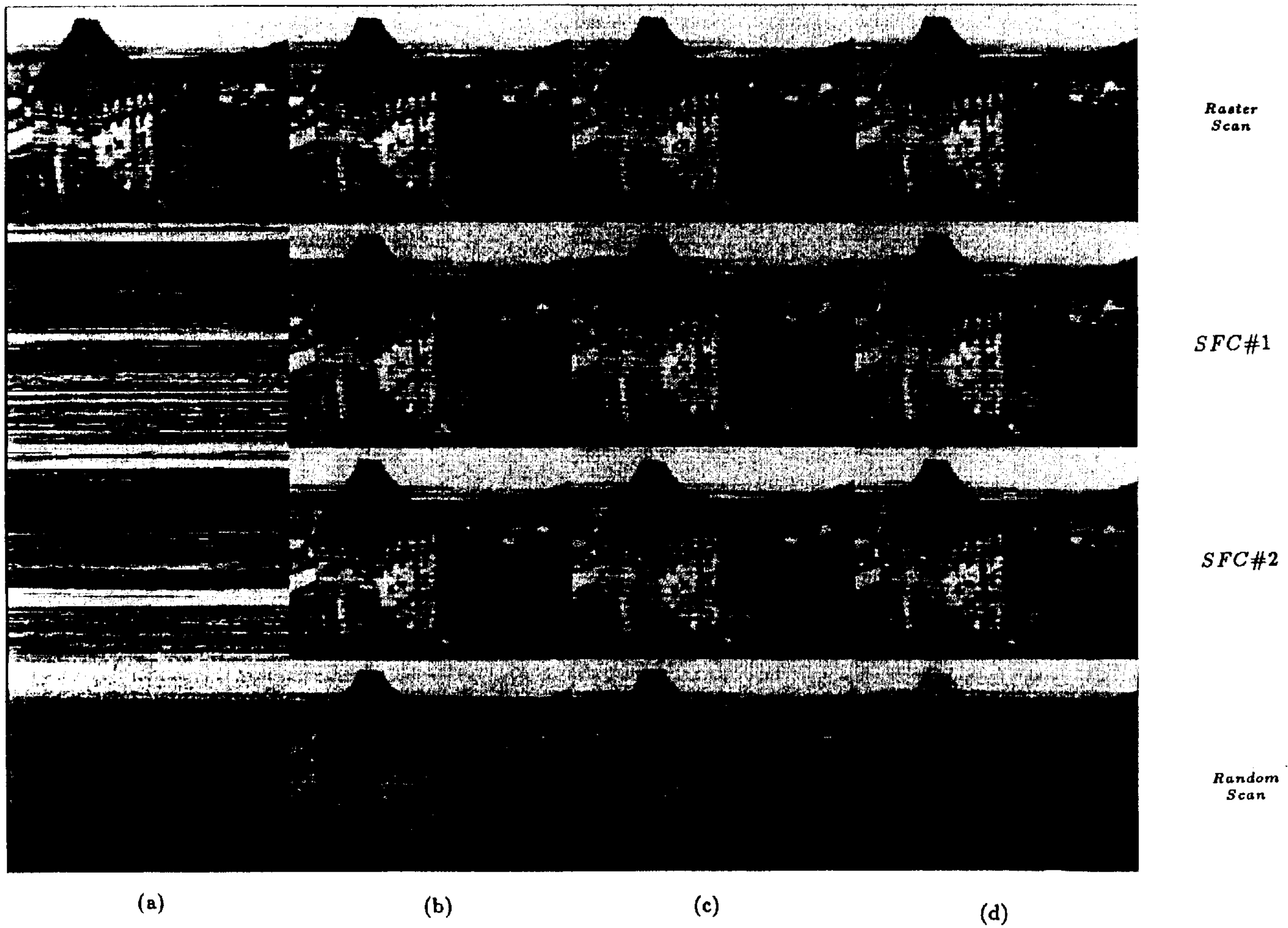


Figure 8. The 4 signals (a), the output of the 80%, 60% *and* 40% *filterings of the Landscape picture's signals (by the 4 scans) in (b), (c) and (d), respectively.*

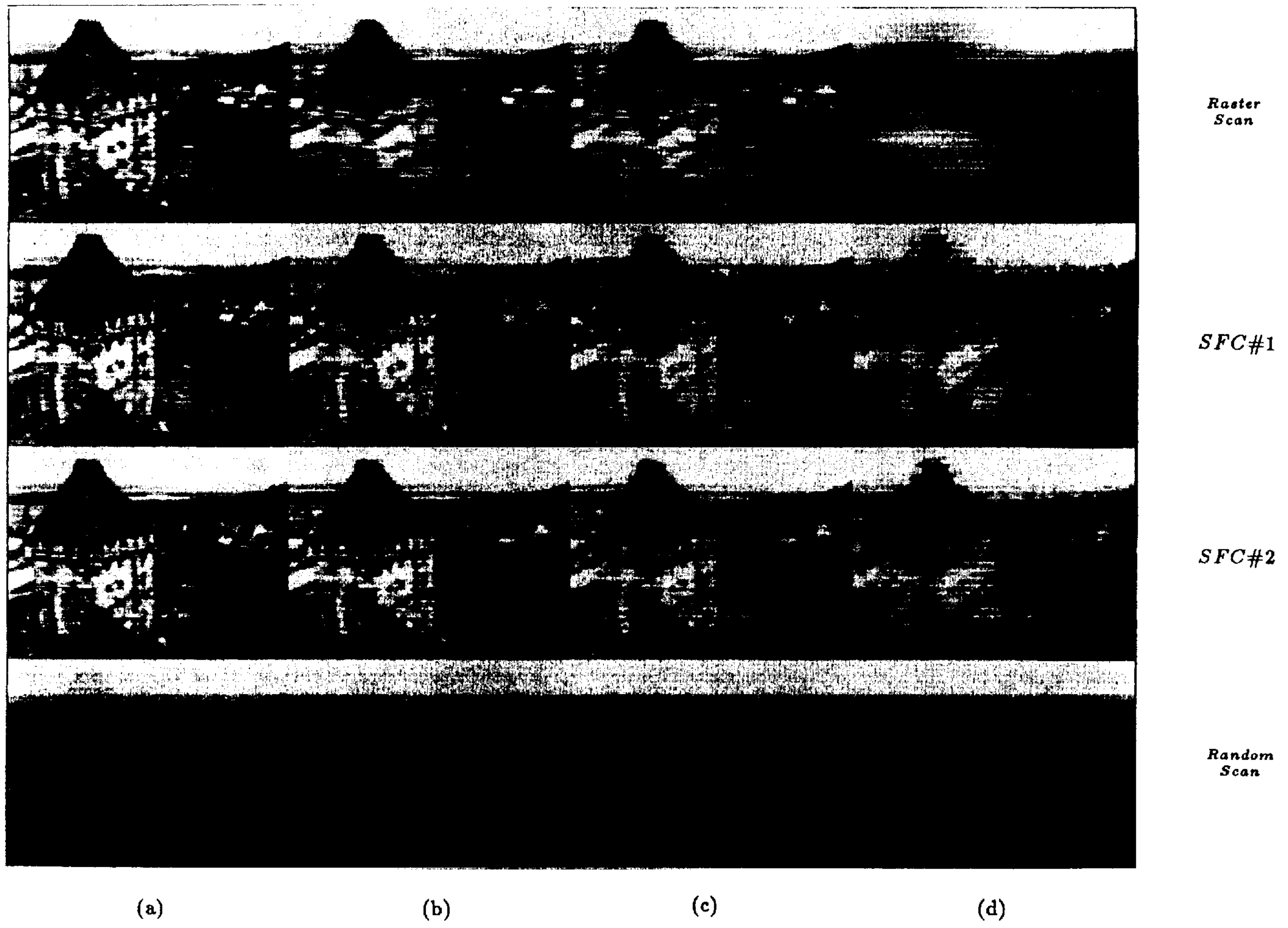


Figure 9: The output of the 20%, 10%, 5% *and* 1% *filterings of the Landscape picture's signals (by the 4 scans) in (a), (b), (c) and (d), respectively.*

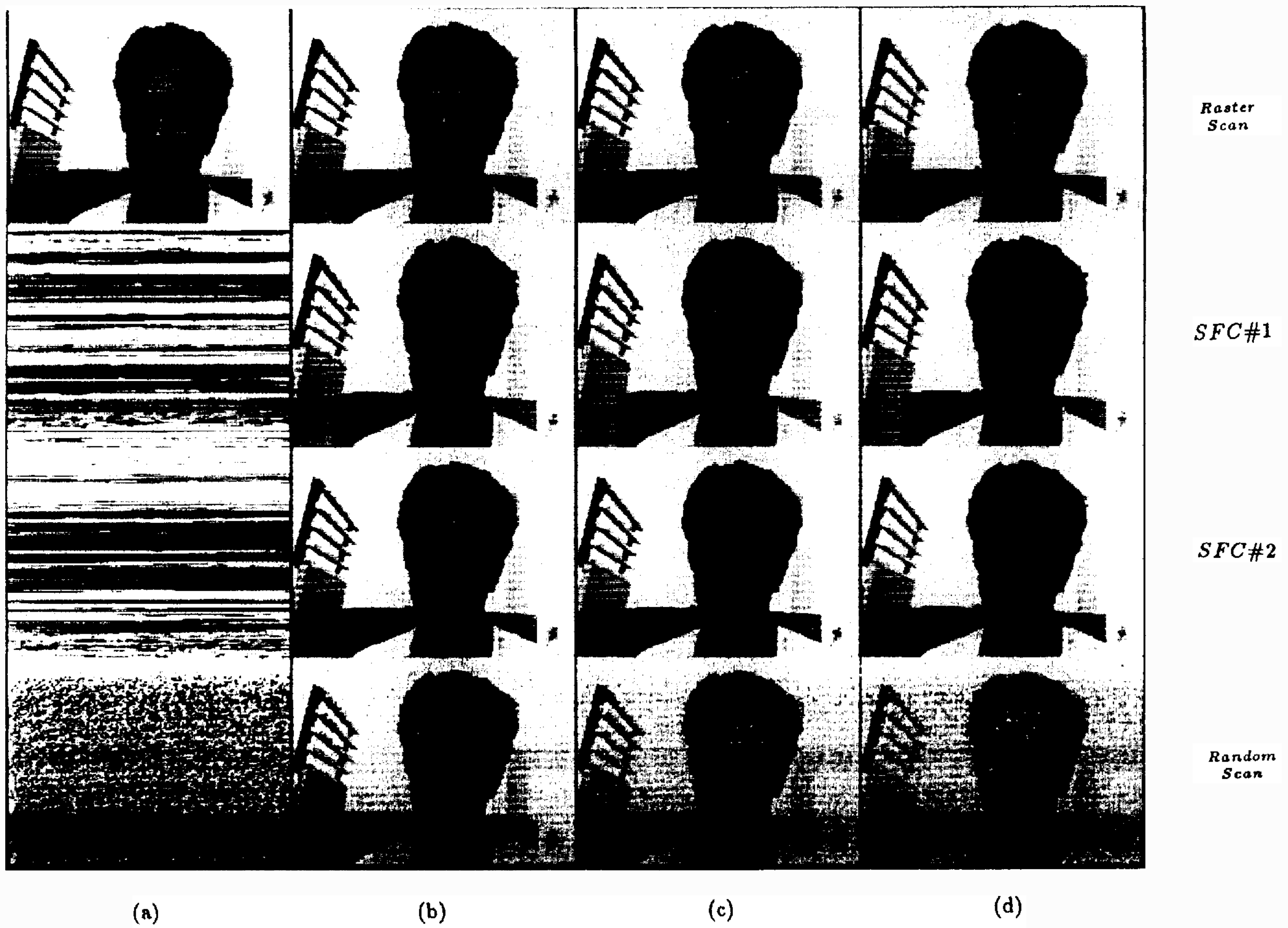


Figure 10. The 4 signals (a), the output of the 80%, 60% and 40% filterings of the Head & Shoulder picture's signals (by the 4 scans) in (b), (c) and (d), respectively.

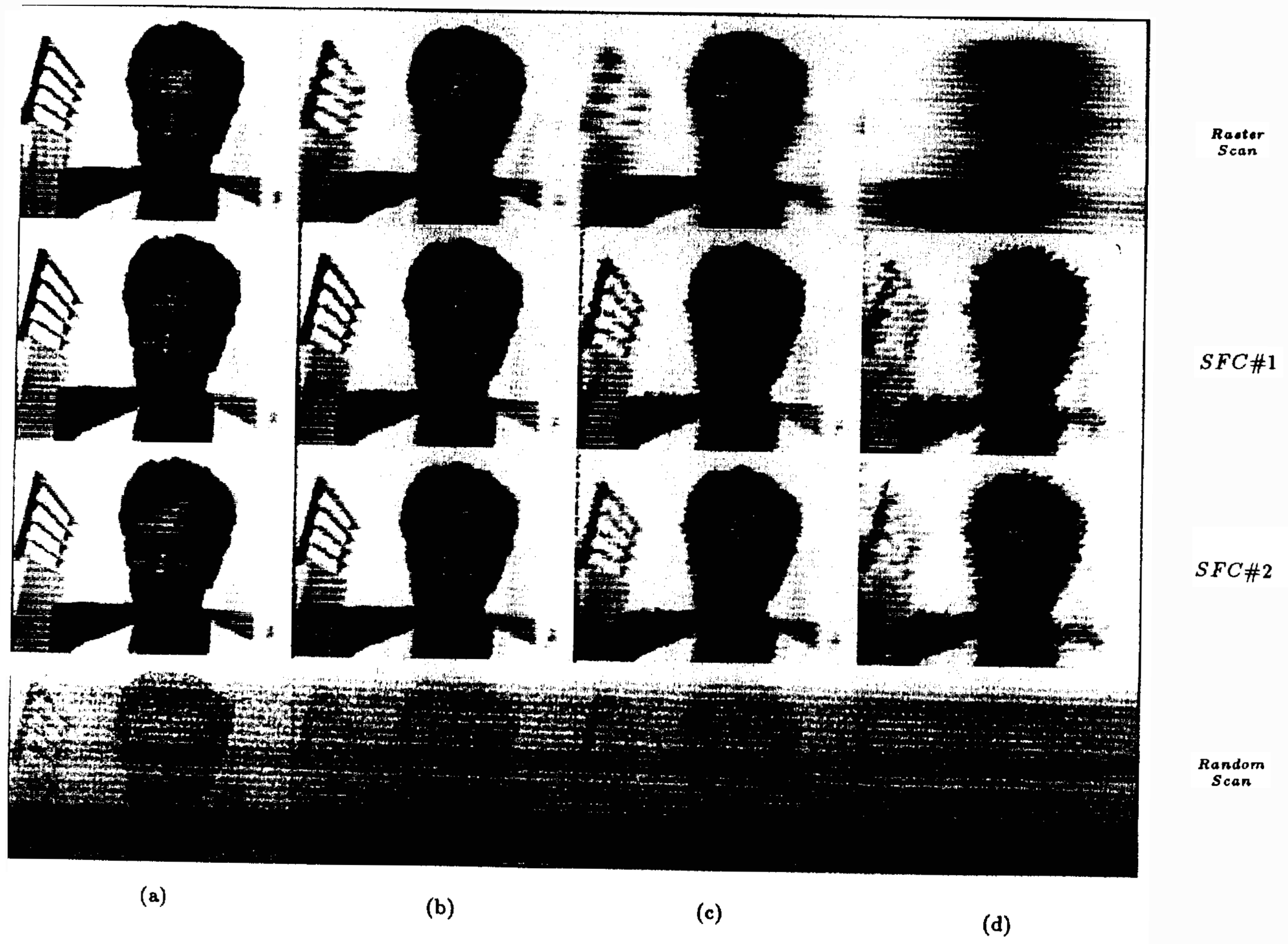


Figure 11. The output of the 20%, 10%, 5% and 1% filterings of the Head & Shoulder picture's signals (by the 4 scans) in (a), (b), (c) and (d), respectively.

In *Fig.* 10.a we can see how the video signals of the Head & Shoulder picture, obtained by the four scans, are displayed on a screen when a descrambler is not used. The filtered Head & Shoulder pictures are shown in *Fig.* 10 b,c,d and in *Fig.* 11.

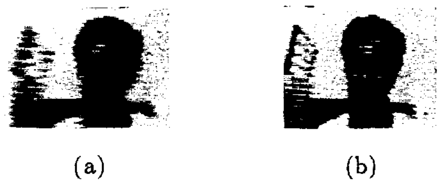


Figure 12. 32 pixels by 32 pixels 5% filtered Head & Shoulder pictures. (a) A Raster Scan's picture. (b) A SFC Scan's picture.

In order to create the effect of looking-from-a-distance we reduced the 5% filtering Head & Shoulder pictures of the Raster Scan and an SFC Scan to 32 pixels by 32 pixels pictures *(Fig. 12)*. Naturally, local disturbances are diminished. We see that the Raster Scan's picture is not recovered; we observe no improvement in the quality due to the reduction in size. On the other hand, the SFC's picture seem to have lost very little information.

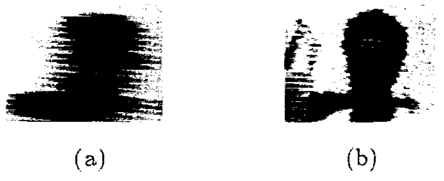


Figure 13. 32 pixels by 32 pixels 1% filtered Head & Shoulder pictures. (a) A Raster Scan's picture. (b) A SFC Scan's picture.

The improvement of the signal by its scrambling is also demonstrated in *Fig.* 13 – the 1% filtering Head & Shoulder pictures of the Raster Scan and of an SFC scan, reduced to 32 pixels by 32 pixels pictures. In the Raster Scan's picture we can only guess that the shadow was originally a Head & Shoulder picture, while in the SFC's picture we can still observe some details.

Finally we give in *Fig.* 14 a closer look at a typical ciphertext produced by the SFC encryption scheme, and challenge the reader to guess what is the corresponding picture.

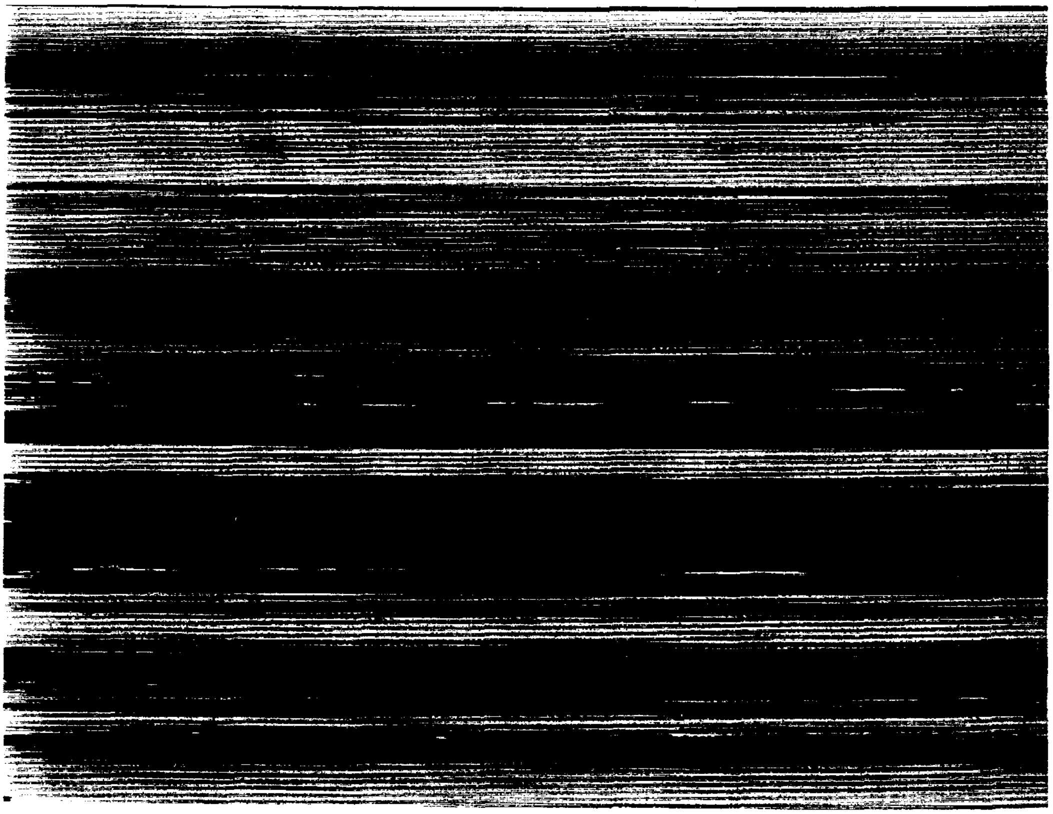


Figure 14.

References

[Fr] – L.E. Franks, "*A model for the random video process*". Bell Systems Tech. J., 45 pp 609-630 (1966).

[IPS] – A. Itai, C.H. Papadimitriou and J.L. Szwarcfiter, "*Hamilton paths in grid graphs*". Siam J. Comput. (1982).

[Kast] – P.W. Kasteleyn, "*Graph theory and crystal physics*". in "Graph theory and theoretical physics". ed. F. Harary, Academic Press, London, 1967.

– *"A soluble self-avoiding walk problem."* Physica, 29, 1329-1337, 1963.

[LZ] – A. Lempel and J. Ziv, *"Compression of two-dimensional data"*. IEEE Trans. Inform. Theory, vol. IT-32, no. 1, pp. 2-8, Jan. 1986.

[NL] – A.N. Netravali and J.O. Limb, *"Picture Coding: A Review"*. Proc. IEEE, vol. 68, no. 3, March 1980.

[OID] – H. Orland, C. Itzykson and C. de Dominicis, *"An evaluation of the number of Hamiltonian paths"*. J. Physique Lett. 46, L-353-L-357, April 1985.

[Ren] – S. Render, *"Distribution functions in the interior of polymer chains"*. J. Phys. A: Math. Gen. 13, 3525-3541, 1980.

[W1] – A.D. Wyner, *"An analog scrambling scheme which does not expand bandwidth, Part 1 : Discrete time"*. IEEE Trans. Inform. Theory, vol. IT-25, pp.261-274, May 1979

[W2] – A.D. Wyner, *"An analog scrambling scheme which does not expand bandwidth, Part 2 : Continuous time"*. IEEE Trans. Inform. Theory, vol. IT-25, pp.415-425, July 1979

SECURE AUDIO TELECONFERENCE

E. F. Brickell, P. J. Lee, Y. Yacobi

Bell Communications Research
435 South Street
Morristown, N.J. 07960, U.S.A.

ABSTRACT

A number of alternative encryption techniques have been suggested for secure audio teleconferencing implementable on public switched network, in which the centralized facility, called *bridge*, does not hold any secret. The role of the bridge is to synchronously add simultaneous encrypted signals, modulo some known number, and then transmit the result to all the participants. Each terminal has a secret key, with which it can decrypt the above modular sum of encrypted signals to obtain the desired ordinary sum of cleartext signals. Secrecy of the systems is analyzed. Some of which are provably secure, assuming the existence of one way functions, and for the others we have partial cryptanalysis.

We also present a N-party identification and signature systems, based on Fiat and Shamir's single party systems, and another N-party signature system based on discrete-log problem. Our systems have communication complexity $2N$ times that of the basic Fiat-Shamir systems (as compared to a factor of N^2 in the direct application of the basic scheme to all pairs).

KEY WORDS: *secrecy*, *identification*, *signature*, *bridging*

I. INTRODUCTION

When holding a teleconference, it is common for all participants to be connected to a central facility called a *bridge*. The bridge receives the signals from the participants, operates on these signals in an appropriate way, and then broadcasts the result to the participants. Let N be the number of participants in the teleconference. We will assume that the bridge can receive and process N signals in parallel.

In an audio teleconference the bridge could just add the speech signals from all the participants and broadcast the sum. This is not preferable because this would also add the background noise from all the participants and would require unnecessarily large dynamic range. Instead, the bridge typically selects only up to L active speakers, adjusts the volume, adds their signals, and broadcasts the result. L is typically quite small, only 2 or 3, but we will keep it variable for the remainder of this paper. (Certain audio bridges limit the number of broadcastable speaker to be one. But for interactive teleconferencing, which is of our interest, L is required to be at least 2.)

In this paper, we consider the problem of holding a *secure* audio teleconference. The participants will encrypt their digitized speech in the form of linear and non-adaptive PCM or DPCM. The bridge will operate on the encrypted signals without any decryption and broadcast the result. The participants must be able to decrypt and recover the desired signal.

Since from the incoming encrypted signal the bridge cannot determine the activity, the transmitter of each user will determine its own activity and send encrypted speech to the bridge only when they are actually active. (When not active, the transmitter sends a cleartext "idle" signal to the bridge if circuit switched network is used or does not send anything if packet switched network is used.) However if there are more than L participants speaking at a given time, the bridge must decide which L signals to use.

In the initialization phase of the conference, each participant will probably want to be convinced of the identity of each of the other participants. Then they will establish cryptographic keys that will be known to all of the participants, but must be unknown to any outside observer, including the bridge.

We would also like to have a mutual signature of all participants on the contents of the conference. Signatures might also be needed if the participants wish to sign a mutual document during the conference.

In section 2 we present some secure audio teleconferencing systems. Decryption by the participants of the combined signal results in the sum of clear individual signals. The required increase in message size equals (in bits) the logarithm of the maximum number of simultaneously active speakers allowed. The solutions differ in cryptographic strength and system complexity. All the solutions require a mutual synchronous time base in all the terminals and in the bridge.

Secure audio bridges are related to the idea of privacy homomorphisms [RAD]. Privacy homomorphisms would not require synchronous operation of the terminals. However, in [BY], it is shown that two of the additive privacy homomorphisms mentioned in [RAD] are insecure against a ciphertext only attack, and the other two are insecure against a known plaintext attack.

In section 3 we present a N-party identification system and signature system. These systems are based on Fiat and Shamir's 2-party systems, which are much more efficient than using the RSA system for identification or signature. Our systems have communication complexity about 2N times that of the basic systems (compared with N^2 in the direct application of the basic system to all pairs). These protocols may be parallelized easily, unlike the direct application of the basic Fiat-Shamir scheme to all pairs. For example, the identification protocol, when implemented in parallel, takes just twice the time of the basic 2-party protocol, independent of the number of participants.

II. SECURE AUDIO TELECONFERENCE SYSTEMS

2.1 System Descriptions

The following notations all refer to the message (quantized analog value), cryptogram etc., at instant (sample time) t, therefore we omit the indication t. Let the message space M be the integers $0,1,2, \cdots ,B$. The cipher space C will be the integers mod P for some $P > L \cdot B$. Let M_i and C_i denote the message and cryptogram of participant i (at time t). To simplify notation, we assume that participants $1,2,...,Q$ are the Q active (allowed) speakers $(Q \leq L)$. f denotes an easily computable function, which is hard to invert. f will produce a random integer *mod* P from a key and a sync word (t may be used as a sync word).

(a) Distinct key / Common Sync Additive System

Encryption of message M_i is $C_i = f_{K_i}(t) + M_i \bmod P$. The bridge computes $C_T = \sum_{i=1}^{Q} C_i \bmod P$ and broadcasts it to all the participants. Decryption is done by subtracting the corresponding sum $\sum_{i=1}^{Q} f_{K_i}(t)$ *mod* P, and the result is $M_T = \sum_{i=1}^{Q} M_i$ *mod* P. The bridge needs to notify the receivers of the user IDs of the Q active speakers. This can be very infrequent (say once every 10 ms). The set of keys of all the participants must be known to each of them.

(b) Common Key / Common Sync Additive System.

For this system, the running keys for all the participants are the same since they use a common key K and common sync. The bridge broadcasts C_T as well as (infrequently) the value of Q. The receiver subtracts $Q \cdot f_K(t)$ from C_T to obtain M_T. This system is not as secure as the previous one as we will show shortly, but it greatly simplifies the hardware of the receiver, reduces the amount of side information (just Q, not IDs), and reduces the initial start-up duration since it needs to distribute only one key.

(c) Common Key / Distinct Sync Additive System

If the transmitter uses its ID as a part of the sync word then the resulting running keys are all different. Hence, under some assumptions on the function $f(\cdot)$, this system is as secure as the distinct key system. The bridge broadcasts C_T as well as the active users IDs (infrequently). The receiver subtracts $\sum_{i=1}^{Q} f_K(i;t)$ from C_T to get M_T. Its hardware is more complex than that of system (b), but in some implementations may be less complex than that of system (a).

(d) Common Key / Common Sync Multiplicative System

In this system $C_i = M_i \cdot f_K(t) \bmod P$. We will assume that $f_K(t)$ has an inverse modulo P. The bridge sums the cryptograms mod P, and decryption is done by multiplying the total cryptogram by $f_K(t)^{-1}$ *mod* P, to result M_T.

The following Lemma holds for all systems.

Lemma 2.1 : $M_T = \sum_{i=1}^{L} M_i$.

Proof : $M_T = \sum_{i=1}^{L} M_i$ *mod* P, and $\sum_{i=1}^{L} M_i \leq L \cdot (B) < P$.

2.2 The Security of the systems

We assume that the function $f_K(t)$ is a pseudo random function chosen from a poly-random collection [GGM]. This means that Knowing the output of f on polynomially many inputs one cannot infer, using polynomially bounded resources, the output for some other input with probability significantly higher than guessing. Such functions exist if one way functions exist (i.e., easily computable functions, which are hard to invert on nonnegligible portion of their target) [GGM]. In real applications we believe that using DES as the function $f_K(t)$ is sufficient, however, to prove the following three theorems we need the assumption that f is a pseudo random function.

Theorem 2.2.1 : All the systems are secure for a single speaker.

Sketch of Proof : Using Shannon theory, they are secure for a single speaker if the key sequence is truly random. If we replace the truly random key sequence with a pseudo random function then the system is secure because a proof of its insecurity will establish an efficient way to distinguish between the pseudo random function and a truly random function [GGM].

Theorem 2.2.2 : Systems (a) and (c) are secure for multiple speakers.

Sketch of Proof : The outputs of the pseudo random functions for systems (a) and (c) are all *distinct* and unpredictable (within polynomial resources) for all the users. Hence, if there is cryptanalysis for these systems with multiple speakers, then there is also cryptanalysis for a single speaker, which contradicts the previous theorem.

The outputs of the pseudo random function for systems (b) and (d) are not distinct. At each time instance all the N generators produce the same output. Therefore we cannot prove a theorem similar to 2.2.2 for these systems. In fact Theorem 2.2.3 states the contrary.

Theorem 2.2.3 : Systems (b) and (d) are partially insecure for multiple speakers.

Sketch of Proof : Recall that $B<P/L$. For a given cryptogram C, the probability for a pseudo random key R to decipher C into a value in M is B/P. Therefore, the probability for a R to decipher Q different cryptograms simultaneously into values in M is $(B/P)^{-Q}$. Since there are P pseudo random keys to try, the number of possible solutions is $P\cdot L^{-Q}$ on the average. In addition, in system (b), if there are more than two speakers, then the cryptograms which correspond to the largest and the smallest M's can be identified. Hence with probability $\approx 3/4$, the most significant bits of the two extreme messages can be cryptanalyzed.

III. IDENTIFICATION AND SIGNATURE IN TELECONFERENCING

3.1 Introduction

Fiat and Shamir [FS] invented a single-party identification scheme and a signature scheme, based on the ideas of *zero knowledge proofs* These schemes are secure if factoring is hard. The implementation is more efficient than RSA.

For an N-party teleconference we present similar systems with communication complexity about $2N$ times that of the basic systems (instead of N^2 in the direct application of the basic systems to all pairs).

These protocols can be implemented with high parallelism, and therefore, if computation time is negligible compared to communication time, the identification process of an N-party conference takes about twice the time of 2-party identification protocol, independent of N. Recall that in our model, we assume that the bridge can receive and process N signals in parallel.

Our protocols do not solve the simultaneity problem, i.e. Some parties may quit the process after receiving the signatures of others, and before delivering theirs. If these signature protocols are used to sign the conference itself (say each second), then simultaneity is not important, because the honest parties will quit the conference, as soon as anybody stops cooperating. However, simultaneity is important for documents signature, hence for this purpose our protocols are inappropriate.

In [KO] Koyama and Ohta propose a N-Party identity based key distribution system. Their system provides identification and key-distribution, while our system provides identification and signature.

In addition, we give an N-party signature protocol, based on discrete-log problem. (N-party discrete-log identification protocols were published by Chaum and Van de Graaf [CH].)

3.2 Detailed Description

Each user u has some unique identifying data I_u associated with him (e.g. name, address, social security number). This data is not secret.

Let h be any cryptographically secure pseudo random function. For a more precise definition see [GGM]. Let k be an integer. Everybody can compute $v_{uj}=h(I_u,j)$, $j=1,2,3,..,k$. Without loss of generality we assume that $v_{u1},...,v_{uk}$ all have Jacobi symbol +1 (rename the first k v_{uj}, which have Jacobi symbol +1). Everybody can efficiently compute the Jacobi symbol of every number.

In this section all congruences are modulo n, the product of two large primes, which are congruent to 3 modulo 4. (Such n is known as "Blum-integer".) The nice property of such n is that for every a, if a has Jacobi symbol +1 modulo n, then exactly one of a or $-a$ is a quadratic residue modulo n. We assume that only the central authority knows the factorization of n. Recall that N is the number of users. For user u, the central authority gives u $s_{u1},.....,s_{uk}$ such that for j=1,...,k, $s_{uj}^2 \equiv \pm v_{uj}^{-1}$. It is the knowledge of the factorization of n, which enables the central authority to compute square roots modulo n.

We will need to have a one way function g which will map very long messages into a relatively short sequence of bits. Let M be the message to be signed. (M could be the concatenation of all encrypted signals during the last second, or a document.) Let Γ be the list of participants.

3.2.1 The quadratic residue signature protocol

(1) Each participant $1 \leq u \leq N$ picks random $0 \leq r_u \leq n$, computes $x_u \equiv r_u^2$, and sends (x_u) to the bridge,

(2) The bridge computes $X \equiv \prod_{u=1}^{N} x_u$ and broadcasts it.

(3) Each participant u computes $g(M,X,\Gamma) = (e_1,..e_k)$,

(4) Each participant u computes $y_u \equiv r_u \cdot \prod_{j=1}^{k} s_{uj}^{e_j}$, and transmits y_u to the bridge,

(5) The bridge computes $Y \equiv \prod_{u=1}^{N} y_u$, and broadcasts it ,

(6) Each participant computes $Z \equiv Y^2 \prod_{j=1}^{k} V_j^{e_j}$, where $V_j \equiv \prod_{u=1}^{N} v_{uj}$, and $v_{uj}^{-1} \equiv s_{uj}^2$,

(7) Y is a valid signature if and only if $Z \equiv \pm X$, and the list Γ of parties matches the list of parties signing the documents.

[]

Remarks:

(1) If nobody cheats the protocol terminates positively.

(2) To show that this signature scheme is secure, we need to show that it is difficult for a set R of users to produce a document that is signed by a set $R \cup S$ of users. We can show that this problem is essentially the same problem faced by a forger in the original FS signature scheme (except for simultaneity).

(3) Communication Complexity: 4N communications of length log n, but the time required is only the time for four communications of length log n.

(4) The list Γ of parties signing the document is used to overcome a problem that arises in mutual signatures that does not occur in single signatures. A participant N+1 who did not sign the original document may at some later date wish to have his signature on the document. He could take the string of bits $e_1, \cdots, e_k$ and form $Y_{n+1} = \prod_{j=1}^{k} S_{n+1,j}^{e_j}$. (He is essentially using $r_{n+1}=1$.) Then $Y' = Y_{n+1}$ would be accepted as a valid signature of N+1 parties if there was no requirement concerning the list Γ.

(5) k is the security parameter. This protocol requires each user u to have a large number $k \approx 100$ of identifiers $S_{u1}, \ldots, S_{uk}$. This number can be reduced to k/τ by a modification that would also increase the communication complexity by a factor of τ.

3.2.2 The quadratic residue identification protocol

One way to achieve identification is to have all participants use the signature protocol to sign the empty message. The function g is used only as a way of generating the bits $e_i, \ldots, e_k$. Each participant wants to be convinced that even if the bridge and all $N-1$ other participants colluded, there is still some randomness in these bits. There are other ways of generating such a string of bits for an identification protocol. One method is to have each participant send in δ bits and then use the concatenation of all of these bits as the string $\bar{e}$. For this $\delta \approx 20$ would probably be sufficient.

3.2.3 Discrete-log N-party signature protocol

We show that discrete-log problem can be used efficiently for signature in N-party teleconference. We could use discrete log either over finite integer fields, or over finite polynomial fields. Operations in GF(2^{1000}) are more efficient then in GF(P), where P $\approx 2^{500}$, while they have about the same security.

Comparing the polynomial discrete-log signature systems, to the quadratic residue signature system, taking in account the current difference between shift register technology and CPU technology, we conclude that the polynomial version discrete-log signature system is almost an order of magnitude faster than the quadratic residue signature system in computation time. (The above comparison is true for ten users, and security parameter $k=50$. The security parameter determines the error probability $=2^{-k}$.) On the other hand it is twice slower in communication time (because the messages are twice longer).

Let T be a prime power, and, as before, let Γ be the list of parties, and M the message to be signed. $\equiv$ denotes congruence in $GF(T)$. Each user u, $1 \leq u \leq N$ has secrets s_{uj}, $1 \leq j \leq k$. $w_{uj} \equiv \alpha^{-s_{uj}}$ is public. To avoid the use of a "telephone book" for public keys, each party can deliver his claimed public key certified by the trusted center. α is a generator of the multiplicative group of the field.

The protocol

(1) u picks random $r_u \epsilon [0,T)$, computes $a_u \equiv \alpha^{r_u}$, and transmits it to the bridge.

(2) The bridge computes $A \equiv \prod_{u=1}^{N} a_u$, and broadcasts it,

(3) Let $(e_1,\ldots,e_k)$ be the first k bits of α^{σ} in $GF(T)$, where σ is the concatenation (M,A,Γ). u computes $y_u \equiv r_u + \sum_{e_j=1} s_{uj} \mod T-1$, and transmits it to the bridge.

(4) The bridge computes $Y \equiv \sum_{u=1}^{N} y_u \mod T-1$, and broadcasts it.

(5) u computes $W \equiv \prod_{u=1}^{N} \cdot \prod_{e_j=1} w_{uj}$, and then $Z \equiv \alpha^{Y} \cdot W$.

(6) Iff $Z \equiv A$, then o.k.

[]

The signature is composed of all the above communications.

REFERENCES:

[BY] E. Brickell and Y. Yacobi, "On Privacy Homomorphisms", **Eurocrypt-87,**

[CH] D. Chaum, J. van de Graaf :" An Improved Protocol for Demonstrating Possession of a Discrete Logarithm and Some Generalizations", **Eurocrypt-87** ,

[FS] A. Fiat and A. Shamir :"How to prove yourself: Practical Solutions to Identification and Signature Problems", to appear in Proc. **Crypto-86.**

[GGM]O. Goldreich, S. Goldwasser and S. Micali:"How to Construct Random Functions", **JACM** , Vol.33, No.4, Oct. 1986, pp.792-807.

[KO] K. Koyama and K. Ohta :"Identity based conference key distribution systems", to appear in **Crypto-87** .

[RAD] Rivest, Adleman and Dertouzos :" On data banks and privacy homomorphisms", in Foundations of secure computation, edited by Demillo et al. **Academic Press** 1978.

[Y] A.C. Yao, "Theory and Applications of Trapdoor Functions", Proc. **Foundations of Computer Science,** 1982, pp.80-91.

ATTACK ON THE KOYAMA-OHTA IDENTITY BASED KEY DISTRIBUTION SCHEME

by

Yacov Yacobi

Bell Communications Research
435 South Street
Morristown, NJ 07960
USA

ABSTRACT

Koyama and Ohta proposed an identity based key distribution scheme. They considered three configurations: ring, complete graph, and star. The most practical configuration is the star which is used in teleconferencing. We show the Koyama-Ohta star scheme to be insecure. Specifically, we show that an active eavesdropper may cut one of the lines, and perform a bidirectional impersonation, thus establishing two separate keys. One with each side.

1. INTRODUCTION

Koyama and Ohta proposed [1] an identity based key distribution scheme. They considered three configurations: ring, complete graph, and star. The most practical configuration is the star which is used in teleconferencing. We show the Koyama-Ohta star scheme to be insecure. Specifically, we show that an active eavesdropper may cut one of the lines, and perform a bidirectional impersonation, thus establishing two separate keys. One with each side.

The same kind of attack is possible for the classic Diffie-Hellman key distribution scheme [2], however, this scheme is not an identity based scheme. Diffie and Hellman do not claim to solve impersonation problems in their scheme.

In order to apply this attack to the complete graph case, the eavesdropper has to manipulate all the communication lines which connect one node to each of the rest. This is much less probable than the attack on the star configuration, which requires the manipulation of just one line.

2. THE SCHEME, [1]

Each of the parties is equipped with a smart card, issued by trusted center. Let p,q and r be large primes, where p and q are secret, known to the center only, and r is public. Let L=lcm(p-1,q-1,r-1). Let e be any (public) number relatively prime to L, and d its inverse modulo L (i.e. $e \cdot d \equiv 1 \ mod \ L$. Let $3 \leq c < L$. Let g be a primitive element of GF(p), GF(q), and GF(r). To make the search for such g practical we demand that p-1=2p', q-1=2q' and r-1=2r', where p', q', and r' are primes.

Let ID_i denote the identification information of user i. The center computes $S_i \equiv ID_i^d \ mod \ nr$, where n=pq. The center stores (n,r,g,e,c,Si) in the smart card of user i. Si,p,q and d are secret.

The Protocol

j picks random $U_j \epsilon [0,n)$, and *sends* $E_j \equiv_n g^{e \cdot U_j}$, to i , (1.1)

i picks random P_i and V_i, *where* $P_i \epsilon [0,nr)$ and $V_i \epsilon [0,n)$, and computes

$$X_i \equiv_{nr} g^{e \cdot P_i}; \ Y_i \equiv_{nr} S_i \cdot g^{c \cdot P_i}; \ Z_{ij} \equiv_n E_j^{P_i}; \ F_i \equiv_n X_i^{e \cdot V_i}. \quad (1.2)$$

i then sends (X_i, Y_i, Z_{ij} and F_i) to j.

j checks whether the following two congruences hold:

$$Y_i^e / X_i^c \equiv_{nr} ID_i \text{ and } Z_{ij} \equiv_n X_i^{U_j}. \quad (1.3)$$

If they do not hold j halts.

j picks random $R_j \epsilon [0,nr)$, computes the following three numbers, and sends them to i:

$$A_{ji} \equiv_{nr} X_i^{e \cdot R_j}; \ B_{ji} \equiv_{nr} S_j \cdot X_i^{c \cdot R_j}; \ C_{ji} \equiv_n F_i^{R_j} \quad (1.4)$$

i checks whether the following two congruences hold:

$$B^{e}_{ji}/A^{e}_{ji} \equiv_{nr} ID_j \text{ and } C_{ji} \equiv_n A_{ji}^{V_i} \tag{1.5}$$

i halts if they do not hold.

i computes conference key

$$K_i \equiv_r A_{ji}^{P_i} \tag{1.6}$$

j computes conference key

$$K_j \equiv_r g^{e^2 \cdot R_j} \tag{1.7}$$

[]

From (1.2), (1.4) and (1.6) it is clear that $K_i \equiv_r K_j$.

3. THE ATTACK

The eavesdropper cuts the communication line between the honest center of the star (j) and one of the terminals (i). He mediates every communication between the two from now on. When communicating with j he pretends to be i (denoted $\tilde{i}$), and when communicating with i he pretends to be j (denoted $\tilde{j}$). At the end of the attack protocol $\tilde{j}$ establishes a key with i, and $\tilde{i}$ establishes another key with j. The key with j matches the session key of the rest of the group.

Preprocessing:

The eavesdropper choses random P', and computes its inverse modulo r-1 ($\overline{P}$'). He also compute the inverse of e modulo r-1 ($\overline{e}$).

The attack protocol

j picks random secret $U_j \epsilon [0, n)$, and computes

$$E_j \equiv_n g^{e \cdot U_j} \tag{2.1}$$

He then sends it to i. $\tilde{i}$ reads it without interfering.

i picks random $P_i \epsilon [0, n\cdot r)$ and $V_i \epsilon [0,n)$, computes

$$X_i \equiv_{n\cdot r} g^{e\cdot P_i};\ \ Y_i \equiv_{n\cdot r} S_i \cdot g^{c\cdot P_i};\ \ Z_{ij} \equiv_n E_j^{P_i};\ \ F_i \equiv_n X_i^{e\cdot V_i}, \tag{2.2}$$

and sends it to j,

$\tilde{j}$ intercepts the message, and modifies it as follows: The new Z_{ij} and F_i equal the original, but the new X_i and Y_i (denoted $\dot{x}_i$ and $\dot{y}_i$ are computed using Chinese Remanidering to have the following properties:

$$\dot{x}_i \equiv_n X_i;\ \ \dot{x}_i \equiv_r g^{e\cdot P}\ \ (denoted\ \ x_i');\ \ \dot{y}_i \equiv_n Y_i;\ \ \dot{y}_i \equiv_r (ID_i \cdot (x_i')^c)^{\bar{e}}; \tag{2.2'}$$

He then sends $\dot{x}_i$; $\dot{y}_i$; Z_{ij}; F_i to j,

j validates that

$$\dot{y}_i^e / \dot{x}_i^c \equiv_{nr} ID_i \ \text{ and } \ Z_{ij} \equiv_n \dot{x}_i^{U_j}, \tag{2.3}$$

j halts if these congruences do not hold.

j computes the following three numbers:

$$\dot{a}_{ji} \equiv_{nr} \dot{x}_i^{eR_j};\ \ \dot{b}_{ji} \equiv_{nr} S_j \cdot \dot{x}_i^{c\cdot R_j};\ \ C_{ji} \equiv_n F_i^{R_j} \tag{2.4}$$

and sends them to i,

$\tilde{i}$ intercepts this communication. He choses some random $\dot{r}_j$. He then modifies the communication as follows: C_{ji} remains unchanged. Using Chinese Remaindering , $\tilde{i}$ computes new $\ddot{a}_{ji}$, and $\ddot{b}_{ji}$, for which the following holds:

$$\ddot{a}_{ji} \equiv_n \dot{a}_{ji} \equiv_n A_{ji};\ \ \ddot{a}_{ji} \equiv_r X_i^{e\dot{r}_j};\ \ \ddot{b}_{ji} \equiv_n \dot{b}_{ji} \equiv_n B_{ji};\ \ \ddot{b}_{ji} \equiv_r (ID_j \cdot X_i^{ce\dot{r}_j})^{\bar{e}}; \tag{2.4'}$$

He sends these three numbers to i.

i verifies that the following congruences hold:

$$\dot{b}_{ji}^{e}/\ddot{a}_{ji}^{c} \equiv_{nr} ID_j \quad ; C_{ji} \equiv_n \ddot{a}_{ji}^{V_i} \tag{2.5}$$

i halts if the congruences do not hold.

i creates session key

$$K_i \equiv_r \ddot{a}_{ji}^{\overline{P}_i} \equiv_r g^{e^2 \cdot \dot{r}_j} \tag{2.6}$$

$\tilde{j}$ creates session key

$$\acute{k}_j \equiv_r g^{e^2 \cdot \dot{r}_j} \tag{2.6'}$$

$\tilde{i}$ creates session key

$$\acute{k}_i \equiv_r \dot{a}_{ji}^{\overline{P'}_i} \equiv_r g^{e^2 \cdot R_j} \tag{2.7'}$$

j creates session key

$$K_j \equiv_r g^{e^2 \cdot R_j} \tag{2.7}$$

$\square$

REFERNCES

[1] K.Koyama and K.Ohta :"Identity based conference key distribution systems", To appear in the proceedings of **Crypto-87** ,

[2] Diffie and Hellman:" New Directions in Cryptography", **IEEE Trans. on Inf. Th.** , **1976.**

On the F-function of FEAL

Walter Fumy, Siemens AG
Systems Engineering Development, E STE 36
D-8520 Erlangen, West Germany

Abstract

The cryptographic strength of a Feistel Cipher depends strongly on the properties of its F-function. Certain characteristics of the F-function of the Fast Data Encipherment Algorithm (FEAL) are investigated and compared to characteristics of the F-function of the Data Encryption Standard (DES). The effects of several straight-forward modifications of FEAL's F-function are discussed.

Introduction

A (cryptographic) function is called *complete*, if each of its output bits depends on every input bit [5]. A block cipher which is not complete, may be vulnerable to a known plaintext attack [1]. As the example of the *Data Encryption Standard* (DES, [7]) shows, the F-function of a Feistel Cipher needs not to be complete in order to ensure completeness of the block cipher itself. Despite the fact that each output bit of its F-function only depends on 6 input bits, the DES is complete after 5 rounds [6].

Due to the principle of a Feistel Cipher which operates on the two halves L and R of an input block, at least 3 rounds are necessary for its completeness. The internal states of a Feistel Cipher develop in the following way:

initial state:	(L,	R)
after round 1:	(R,	$L+F_1(R)$)
after round 2:	($L+F_1(R)$,	$R+F_2(L+F_1(R))$)
after round 3:	($R+F_2(L+F_1(R))$,	$L+F_1(R)+F_3(R+F_2(L+F_1(R)))$)

A 3 round Feistel Cipher therefore is complete if F_2 is complete and if each of the output bits of F_3 depends on at least one of its input bits and each of the input bits of F_1 affects at least one of its output bits.

A function f exhibits the *avalanche effect*, if an average of one half of its output bits change whenever a single input bit is complemented. Moreover, f shows the *strict avalanche criterion* if each of its output bits changes with the probability of one half, when complementing one input bit [9]. The strict avalanche criterion includes the completeness of f. This property is considered essential for a "good" cryptographic transformation [9]. A random function will also exhibit the strict avalanche criterion.

The *dependence matrix* of a function $f:GF(2)^n \text{-->} GF(2)^m$ is a (nxm) matrix, whose entry $a_{i,j}$ gives the probability that the output bit j of f changes when its input bit i is complemented. The function f is complete if all elements in its dependence matrix have a nonzero value; it exhibits the strict avalanche criterion, if the value of every element is close to 0.5.

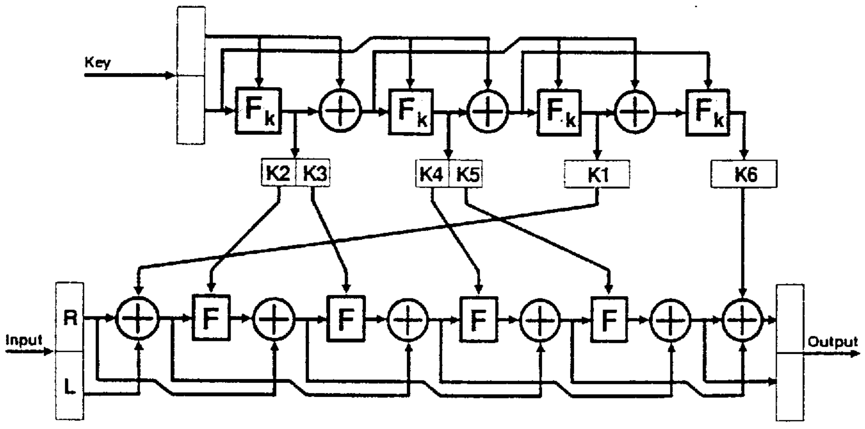


figure 1: Fast Data Encryption Algorithm (FEAL1)

The Fast Data Encipherment Algorithm

The *Fast Data Encipherment Algorithm* (FEAL) is a 6 round Feistel Cipher which operates on 64-bit blocks. Only 4 of its rounds make use of a non-trivial F-function (figure 1 shows FEAL1, [4], which slightly differs from the version given in [8]). The F-function of FEAL is shown in figure 2. The function F_K used for key expansion is similar to F. The structure of F is byte-oriented. Its cryptographic strength depends on a non-linear S-function defined by

$S(x,y,\delta) = \mathrm{ROL2}\ ((\ x + y + \delta\ ) \bmod 256\ )$
where: x, y: one byte data; δ: constant (0 or 1)
ROL2: 2-bit rotate left

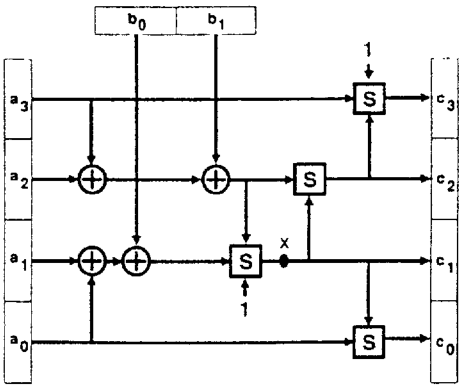


figure 2: F-function of FEAL

Analysis of the S-function reveals its incompleteness. Only one of its 8 output bits depends on every input bit. Moreover, its dependence matrix exhibits a very regular structure. Its upper half (which is identical to the lower half) is given below.

1/64	1/128	1	1/2	1/4	1/8	1/16	1/32		
1/32	1/64	0	1	1/2	1/4	1/8	1/16		
1/16	1/32	0	0	1	1/2	1/4	1/8		
1/8	1/16	0	0	0	1	1/2	1/4		
1/4	1/8	0	0	0	0	1	1/2		
1/2	1/4	0	0	0	0	0	1		
1	1/2	0	0	0	0	0	0		
0	1	0	0	0	0	0	0		

Since the S-function is not complete, the F-function of FEAL can not be complete. An analysis of the powers of F results in F^i being complete for i > 1 and exhibiting the strict avalanche criterion for i > 2 (see figure 4). In this respect the F-function of FEAL does better than the F-function of DES where F^j is complete for j > 2 and shows the strict avalanche criterion for j > 3 (see also figure 4). For this reason FEAL is complete after 4 rounds whereas DES needs 5 rounds in order to become a complete block cipher (see figure 3).

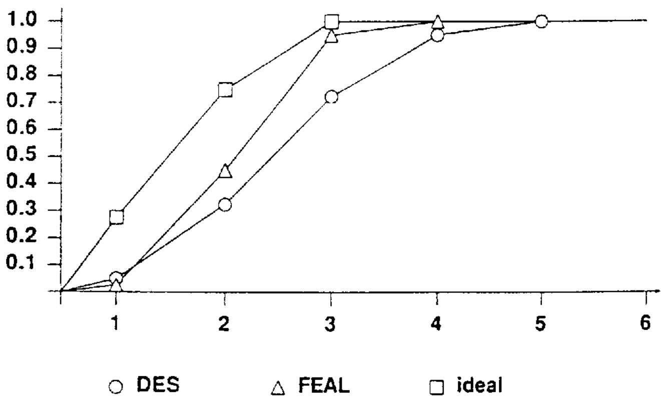


figure 3: Increasing completeness of DES and FEAL

Modifications of FEAL's F-function

There are straight-forward modifications of FEAL's F-function that will further improve the properties discussed above. Moreover, these modifications allow the introduction of a non-linear function that can be chosen at random and will therefore overcome any distrust of the S-functions.

D.Davies has suggested to modify FEAL's F-function by replacing the operation ROL2 by an 8-bit S-box [2]. By this modification 4 (possibly different) S-boxes are added to the F-function. Although a randomly chosen S-Box will show the strict avalanche criterion, the modified F-function will not exhibit the same property. It is complete, however, and F^i will show the strict avalanche criterion for i > 1 (see figure 4).

We suggest a modification that uses only one additional 8-bit S-Box in the position marked by X in figure 2. The input of this S-Box depends on every byte of the input of F. Its output affects every byte of the output of F. The effect of this modification is similar to the effect of the modification discussed above. The modified F-function is complete and F^i will exhibit the strict avalanche criterion for i > 1 (see figure 4).

Both modifications of FEAL's F-function will counteract den Boer's cryptanalysis of this cipher [3].

		F	F^2	F^3	F^4
DES	dependence	0.188	0.875	1.000	**1.000**
	mean	0.119	0.375	0.493	**0.500**
	variance	0.063	0.027	0.00053	**0.00026**
FEAL	dependence	0.883	1.000	**1.000**	**1.000**
	mean	0.308	0.481	**0.500**	**0.500**
	variance	0.112	0.013	**0.00026**	**0.00024**
FEAL + 1 S-Box	dependence	1.000	**1.000**	**1.000**	**1.000**
	mean	0.505	**0.499**	**0.500**	**0.500**
	variance	0.00096	**0.00024**	**0.00026**	**0.00026**
FEAL + 4 S-Boxes	dependence	1.000	**1.000**	**1.000**	**1.000**
	mean	0.503	**0.501**	**0.500**	**0.500**
	variance	0.00054	**0.00025**	**0.00026**	**0.00024**

figure 4: Comparison of different F-functions

References

[1] Chaum, D.; Evertse, J.-H.: *Cryptanalysis of DES with a Reduced Number of Rounds*, in: Advances in Cryptology - Crypto '85, H.C.Williams ed, Lecture Notes in Computer Science, **218** (1986), 192-211

[2] Davies, D.W.: private communication (1987)

[3] den Boer, B.: *Cryptanalysis of FEAL*, presented at Crypto '87

[4] ISO: *Introduction to a New Encipherment Algorithm FEAL*, ISO/TC97/SC20/WG1 N36 (1985)

[5] Kam, J.B.; Davida, G.I.: *Structured Design of Substitution-Permutation Encryption Networks*, IEEE Trans. Computers, **28** (1979), 747-753

[6] Meyer, C.H.; Matyas, S.M.: *Cryptography: A New Dimension in Computer Data Security*, (John Wiley & Sons, New York, 1982)

[7] National Bureau of Standards: *Data Encryption Standard*, FIPS Publ. 46, Washington D.C., 1977

[8] Shimizu, A.; Miyaguchi, S.: *Fast Data Encipherment Algorithm FEAL*, presented at Eurocrypt 1987

[9] Webster, A.F.; Tavares, S.E.: *On the Design of S-Boxes*, in: Advances in Cryptology - Crypto '85, H.C.Williams ed, Lecture Notes in Computer Science, **218** (1986), 523-534

Patterns of Entropy Drop of the Key in an S-Box of the DES

(Extended Abtract)

K.C. Zeng, J.H.Yang, Z.T. Dai

Data and Communications Security Center

Graduate School of Academia Sinica

(I)

The S-boxes used in the DES have been looked at in various aspects like non-linearity,propagation characteristics and I/O correlation immunity. In the present work we try to make a cryptographic study of these boxes from a new viewpoint, namely,by investigating the way in which the uncertainty of the 6-bit key vector k which controls the work of an S-box diminishes,when a certain set of distinct plaintext vectors

$$Qr:\ x(1), x(2), \ldots, x(r)$$

put as queries to the box,together with the signals

$$Rr:\ y_1, y_2, \ldots, y_r,$$

appearing in response at a certain output position are assumed accessible to the cryptanalyst.

Such an approach to the problem of evaluating the S-boxes seems natural and necessary. As a matter of fact,the job of a cryptanalyst consists in nothing else than looking for observable query-response processes as inlets into the secrecy of a system,and in our case the query vectors are observable in reality,at least on the last round of the encryption algorithm.

Since every S-box is composed of four permutations

$$\pi_0, \pi_1, \pi_2, \pi_3$$

acting on the set

$$Z_{16}: 0, 1, 2, \ldots, 15,$$

it suffices to consider the problem for such "simplicial" S-boxes only. Here the vectors k and x belong to $V_4(F_2)$ and we have

$$y=f(x+k),$$

where the output function f(x) at any given position can be expressed in a unique way as a polynomial in $F_2[x_0,x_1,x_2,x_3]$, linear in each indeterminate separately. This is a polynomial of total degree at most 3, and balanced in the sense that it assumes the value 0 exactly eight times over $V_4(F_2)$.

Given any set Qr of r distinct queries to a permutation ,viewed as an S-box, we can introduce an equivalence relation in the set K of 16 possible key vectors by defining

$$k_1 \sim k_2 \Leftrightarrow f(x(i)+k_1) = f(x(i)+k_2),\ i=1,2,\ldots,r,$$

and decompose K accordingly into a disjoint union of equivalence classes

$$K = K_1 \cup k_2 \cup \ldots \cup K_s$$

If, by looking at the presumably accessible response signals y_i, $1 \leq i \leq r$, the cryptanalyst succeeds to decide that the key k in work falls in the class K_t, then the uncertainty of k diminishes from 4 bits to $\log_2|K_t|$ bits, resulting in $4-\log_2 |K_t|$ bits of entropy drop. Thus the average entropy drop of the key which a given query set Qr may cause is

$$C(Q\ ) = -\sum_{t=1}^{s} \frac{|K_t|}{16} \log_2 \frac{|K_t|}{16}$$

and a greater significance in evaluating the cryptographic strength of the permutation π is carried by the parameter

$$C_r = \max \{ C(Q_r) \}.$$

But one must have in mind,that the queries we are talking about here are of an in-active nature. Namely, one can observe which queries have been put to the box, but cannot organize them purposefully. So one has to take into consideration yet an other parameter, namely the probability

$$p_r = \text{Prob } (C(Q_r)=C_r),$$

assuming all the Q s equally possible.

We note in pass that the parameters C_r, p_r can as well be computed for an abitrary polynomial in $F_2[x_0,x_1,x_2,x_3]$.

DEFINITION. The sequence of parameter pairs

$$(C_1 ,p_1), (C_2 ,p_2), \dots, (C_{16} ,p_{16})$$

computed at a given output position of a permutation is called the entropy drop pattern (EDP) of the working key at that position, or to meet the taste of an algebraist,the EDP of the corresponding output polynomial f(x).

THEOREM 1. The EDP of an arbitrary polynimial f(x) in $F_2[x_0,x_1,x_2,x_3]$ is invariant under the group G of 4-dimensional affine transformations acting on the indeterminates,extended by the involutional mapping $f(x) \longmapsto f(x)+1$.

THEOREM 2. The balanced polynomials of $F_2[x_0,x_1,x_2,x_3]$,linear in the indeterminates separately but non-linear in total,form three equivalent classes under the action of the extended affine group G defined above,with class representatives

$$f_1(x)=x_0x_1+x_2,\ f_2(x)=x_0x_1x_2 + x_3,\ f_3(x)=x_0x_1x_2 + x_0x_3 + x_1$$

and corresponding EDPs as tabulated in the following

A			B			C		
$X_0X_1 + X_2$			$X_0X_1X_2 + X_3$			$X_0X_1X_2 + X_0X_3 + X_1$		
R	CR	PR	R	CR	PR	R	CR	PR
1	1	1	1	1	1	1	1	1
2	2	0.80	2	1.81	0.93	2	2	0.60
3	3	0.45	3	2.54	0.80	3	3	0.11
4	3	0.77	4	3	0.61	4	3.5	0.35
5	3	0.92	5	3.41	0.015	5	4	0.11
6	3	0.98	6	3.75	0.22	6	4	0.40
7	3	0.99	7	4	0.09	7	4	0.68
8	3	1	8	4	0.30	8	4	0.84
9	3	1	9	4	0.56	9	4	0.92
10	3	1	10	4	0.78	10	4	0.97
11	3	1	11	4	0.92	11	4	0.99
12	3	1	12	4	0.98	12	4	0.998
13	3	1	13	4	1	13	4	1
14	3	1	14	4	1	14	4	1
15	3	1	15	4	1	15	4	1
16	3	1	16	4	1	16	4	1

We see from the above table that output positions of entropy drop pattern C are more leaky for the key in work than those of pattern B, and positions of patern A are the most unleaky ones,though weaker in the aspect of non-linearity of the output function. This weakness, however, should not be taken too seriously in view of the iterative character of the encryption algorithm as a whole.

At the suggestion of this theorem we computed the 128 EDP's for the 32 permutations used in the DES with the following results

```
  BOX0        BOX1        BOX2        BOX3
 C C C C     C B A C     A C C B     C C C C
 C C C C     A C C B     A B C C     C C C C
 C C C C     B C B A     B C A B     C C C C
 A C A C     C C C C     B C C A     C C C C

  BOX4        BOX5        BOX6        BOX7
 C C A B     C C C C     C B A C     B C C C
 C C C C     C C C C     C B C A     A C B C
 C B C C     C B B C     A C C B     C B B A
 C B C B     A C B C     C A C B     C C A C
```

In addition to the uncanny appearance of Box 3, we have the following disapproving statistics with regard to the choice of the S-boxs in general. It turns out that more leaky patterns occur much more frequently in these boxes.

Pattern	Occurences	Frequency
A	18	14.06%
B	24	18.75%
C	86	67.19%

(II)

If the situation disclosed in the above is to some extent a weakness in the DES algorithm, then it is probably because of the fact that the non-linearity requirement has been unduly overstressed in choosing the boxes, while comparatively less attention has been paid to other equally important aspects such as I/O-correlation immunity and entropy drop of the key. In fact,there exist interesting mutual connections between these three aspects which we try to explain in a couple of words more in the following.

If we denote the input signal to and the output signal from a permutation π by x_i and y_j respectively, and define the I/O-correlation coefficient $c_{i,j}$ as

$$c_{i,j} = \text{Prob } (x_i = 0 \mid y_j = 0),$$

then for the 32 permutations used in the DES we have

$$c_{i,j} = 1/2 + h/8, \quad h = -1,\ 0,\ 1.$$

We call permutations satisfying this restriction selected ones,and the second author of the present paper has proved for them the following result [1]

THEOREM 3. If for any permutation π define the degree of I/O-correlation immunity $c(\pi)$ to be the number of $c_{i,j}$ s which are equal to 1/2, and define the degree of non-linearity $d(\pi)$ to be the sum of the numbers of terms of degree 3 in the four output polynomials, then in the case of a selected permutation we have

$$c(\pi) + d(\pi) = 16.$$

This is a result much more refined than a similar one of Siegenthaler. In particular,it follows from this theorem and theorem2 above, that in the case of an I/O-correlation immune permutation, i.e., a permutation π with $c(\pi)=16$,all non-linear output polynomials are of degree 2, and hence of entropy drop pattern A.

I/O-correlation immune and, more generally,selected permutations satisfying the completenees and all requirements labelled in [2] as "Design Criteria", have been determined and classified by the same author. There are total 17433 equivalent classes of selected permutations under the transformation of subgroups of the symmetrical group S_{16} , including 46 equivalent classes of I/O-correlation immune permutations. His results show that there is enough space for constructing S-Boxes with correlation immune permutations exclusively. This will result in liquidating the leaky patterns B and C in the DES algorithm without bringing about substantial influence on the non-linearity and propagation requirements.

References

[1] J.H. Yang. Structure and Cryptographic Evaluation of S-Boxes of DES Type. Unpublished report.

[2] E.F. Brickell, J.H. Moore and M.E. Purtill, Structure in the S-boxes of the DES, Crypt-86.

THE RAO-NAM SCHEME IS INSECURE AGAINST A CHOSEN-PLAINTEXT ATTACK

René Struik
Eindhoven University of Technology
Eindhoven, the Netherlands

Johan van Tilburg
PTT / Dr. Neher Laboratories
St. Paulusstraat 4
2264 XZ Leidschendam, the Netherlands

ABSTRACT

The Rao-Nam scheme is discussed and generalized to F_q. It is shown that the scheme is insecure against a chosen-plaintext attack for practical code lengths. Based on observations an improved scheme is given, which is not vulnerable to the chosen-plaintext attacks as described.

1. INTRODUCTION

In 1978 McEliece [1] proposed a public-key cryptosystem based on the theory of linear algebraic codes. His scheme was a natural consequence of a paper by Berlekamp, McEliece and van Tilborg [2] in which it was proved that the general decoding problem for linear codes is NP-complete. The McEliece scheme is based on a class of Goppa-codes, which is an extension of the well-known class of BCH-codes. Since there exists fast decoding algorithms for these codes, data rates of 1 Mbits/s [1][3][4] can be obtained. Adams and Meijer [3] and Jorissen [4] computed the optimal values for the parameters of the McEliece system. The optimal values improves the cryptanalytic complexity and information rate of the system. Moreover Adams and Meijer showed that the existence of more than one trapdoor in the McEliece scheme is unlikely.

It is a well-known fact that public-key cryptosystems can be used as private-key cryptosystems. Therefore throughout the years Jordan [5], Rao and Nam [6] proposed to modify the McEliece scheme in order to use it in a classical way. Rao and Nam's aim was to increase the information rate and speed by keeping the generator matrix secret and by using simple error-correcting codes. They modified the McEliece scheme and used the error-correcting properties of the code to determine pre-defined error patterns. The error patterns used in the Rao-Nam scheme have an average Hamming weight of half the code length.

Rao and Nam [6] claimed that the determination of the encipher matrix in the modified scheme from a chosen-plaintext attack has a work factor of at least $T = \Omega(n^{2k})$. Based on the given attack it is suggested in their conclusion that the Rao-Nam scheme for private-key cryptosystems requires only simple codes such as Hamming codes with minimum distance 3 and 4 and is even computationally secure for small $k \gtrsim 50$.

However as will be shown in this paper their exists better attacks from which one can conclude that the Rao-Nam scheme is insecure against a chosen-plaintext attack for practical code lengths. Based on observations we will give an improved scheme which is not vulnerable to the chosen-plaintext attacks as described.

In section 2 we will describe the Rao-Nam scheme for the general case $\mathbb{F}_q$. In section 3 the basic facts about the attack are given. In section 4 Hin's attack on the Rao-Nam scheme with adjacent errors is discussed. The generalized attack by Struik is described in section 5. A ciphertext-only attack, which makes use of an estimated encipher matrix obtained from a chosen-plaintext attack, is given in section 6. In section 7 an improved scheme is given. Finally in section 8 the results obtained are discussed and conclusions are drawn.

2. The Generalized Scheme

The Rao-Nam scheme as described in [6] is a binary scheme. Therefore we shall first generalize this scheme to $\mathbb{F}_q$. For q = 2 the original Rao-Nam scheme is obtained.

Let G denote a (kxn) generator matrix for an [n,k,d]-code $\mathscr{C}$ over $\mathbb{F}_q$ with minimum distance d, dimension k and parity check matrix H. The encryption matrix E is combinatorically equivalent to the generator matrix G and is constructed as follows:

$$E = SGP,$$

where

S is a (kxk) non-singular matrix over $\mathbb{F}_q$ and

P is an (nxn) permutation matrix over $\mathbb{F}_q$.

A message $\underline{m} \in (\mathbb{F}_q)^k$ is <u>encrypted</u> into the ciphertext $\underline{c} \in (\mathbb{F}_q)^n$ as follows:

$$\underline{c} = \underline{m}E + \underline{z}P = (\underline{m}SG + \underline{z})P,$$

where

$\underline{z} \in (\mathbb{F}_q)^n$ is an error vector with an average Hamming weight $W_H(\underline{z}) = \frac{q-1}{q}\, n$.

The matrices S, P and G form the secret key.

The choice of the error vector $\underline{z}$ is to prevent a chosen-plaintext attack by majority voting for each position of a row of the encipher matrix E in the McEliece scheme. If the error vectors have an average Hamming weight $\frac{q-1}{q}$ n the probability of estimating the correct matrix E is on average less than q^{-nk}. Obviously unique decoding is not possible without modifying the original scheme. Therefore Rao and Nam proposed two methods to realize unique decoding for which we have generalized the second method only.

<u>Method 1</u>. Use q=2 and i adjacent errors (ATE) for $\underline{z}$.
An ATE is a vector of length n with i adjacent errors, i.e. an ATE consists of n-i 0's and i consecutive 1's.

<u>Method 2</u>. Use a pre-defined set of error vectors (syndrome-error table).
A pre-defined set of error vectors, consisting of one vector from each coset of the standard array decoding table can be used for $\underline{z}$. Each

coset has a distinct syndrome. Therefore, we can select any set of vectors consisting of one from each of the q^{n-k} cosets. This set of pre-defined errors is part of the secret key.

It is important to note that due to the restricted set of error patterns the system is not optimally secure against a chosen-plaintext attack based on majority voting. For example, if majority voting is used in the Rao-Nam scheme using method 1, then one can use a majority vote with context. Since the number of different error patterns used is just a fraction of the possible number of error patterns we are always better off.

Decryption is straightforward. An enciphered message $\underline{m}$ is <u>decrypted</u> by the following steps.

1) Calculate $\underline{c}' = \underline{c}P^T = \underline{m}SG + \underline{z}$.
2) Determine $\underline{c}^* = \underline{c}'H^T$ and obtain the error pattern $\underline{z} \in (\mathbb{F}_q)^n$.
   As result $\underline{c}'' = \underline{c}' - \underline{z} = \underline{m}SG$ is obtained.
3) Calculate $\underline{m} = \underline{c}''(SG)^{-R}$, in which $(SG)^{-R}$ is a right inverse of the matrix (SG). The result is the plaintext $\underline{m}$.

In the attacks to be described we make use of an equivalent decoding algorithm. The decryption matrix D is HP, since $ED^T = (SGP)(HP)^T = 0$. Note that S is used before the coding process, therefore S has no impact on the error correction. The decoding algorithm is now as follows.

1) Determine $\underline{c}^* = \underline{c}D^T$ and obtain the permuted error pattern $\underline{z}P \in \mathbb{F}_q^n$.
   As result $\underline{c}' = \underline{c} - \underline{z}P = \underline{m}E$ is obtained.
2) Calculate $\underline{m} = \underline{c}'E^{-R}$, in which E^{-R} is a right inverse of the matrix E. The result is the plaintext $\underline{m}$.

3. Weaknesses of Rao-Nam Scheme

The three attacks which will be described make use of the same weaknesses of the Rao-Nam scheme.

The first weakness is the low number of different syndromes, which is for the proposed Hamming code using ATE's at most n, and for the syndrome-error table q^{n-k}. If the number of different error patterns is N, then one has to encipher on average $N(\frac{1}{N} + \frac{1}{N-1} + .. + 1) = \theta(N \log N)$ times to obtain all error patterns. Observe that the minimum distance of the [n,k,d]-code $\mathcal{C}$ plays at this moment no role.

Let $R = \frac{k}{n}$ be the information rate. The number of cosets is $N = q^{n-k} = q^{n(1-R)}$. Consequently in the Rao-Nam scheme there exists a trade-off between information rate and security. For a high information rate R and a large number of cosets N the code length n will be impractical.

The second weakness is due to the leakage of information about the permutation matrix P if ATE's are used. An ATE and its one position shifted ATE (which is an ATE also) differ on exactly two places. After the permutation they still differ on two places. Since we know the structure of the original ATE we can estimate the permutation matrix.

The third weakness is the possibility of estimating the rows of the encipher matrix E=SGP by means of constructing unit vectors $\underline{u}_i$ (= 0..010..0, i.e. the all zero vector with a 1 added on the i-th position). Therefore as suggested in [9] the linear transformation S should be replaced by a non-linear transformation.

4. Attack by Hin

In Hin [7] an attack on the Rao-Nam scheme using ATE's is described. As ATE's have a fixed and known structure, his approach makes use of the leakage of information about the permutation matrix P. In addition it is necessary that the permutation matrix P must transform an ATE into a non-ATE. We will describe his attack for non-cyclic ATE's only.

Let $\mathcal{A}$ denote the ordered set {11...10..0, 011...10..0, ..} of all possible ATE's. The unkown set of permuted ATE's is indicated by $\mathcal{P}$. Let $\mathcal{P}^{(0)}$ be the set of all possible encipherments of the message $\underline{m}=\underline{0}$. An ATE and its one position shifted ATE (which is an ATE also) differ on exactly two places. If the ATE's are distinct and not succesive, then they differ on more than two places. This also holds after permutation and consequently the set $\mathcal{P}^{(0)}$ can be ordered in the same way as $\mathcal{P}$. Hence from the ordered set $\mathcal{P}^{(0)}$ the permutation matrix P is constructed.

Next an ordered set $\mathcal{P}^{(i)}$ is obtained by enciphering the unit vector $\underline{u}_i$ at least N times until all possible error patterns have been appeared, with $1 \leq i \leq k$. The elements in $\mathcal{P}^{(i)}$ are arranged in such a way that the same permutation matrix P is obtained. The x-th elements of each of the sets in $\mathcal{P}^{(0)}$, $\mathcal{P}^{(1)}$, $\mathcal{P}^{(2)}$, ..., $\mathcal{P}^{(k)}$ possess the same error patterns since the sets are ordened in a unique way. Next, with $1 \leq i \leq k$, take the first element in $\mathcal{P}^{(0)}$ and subtract it from the first element in $\mathcal{P}^{(i)}$, i.e.: $(\underline{u}_i E + \underline{z}P) - \underline{z}P = \underline{u}_i E = \underline{e}_i$ which is the i-th row $\underline{e}_i$ of E. In this way the encipher matrix $E = (\underline{e}_1^T, \underline{e}_2^T, \ldots, \underline{e}_k^T)^T$ is constructed. Finally the decipher matrix D, the matrix E^{-1} are calculated and the syndrome-error table is constructed. In this way an (equivalent) decoding algorithm is obtained and the scheme is broken.

SUMMARY

1. Encipher the message $\underline{m} = \underline{0}$ as long as all the N error patterns have not yet appeared.

2. Order the (permuted) error patterns and obtain $\mathcal{P}^{(0)}$. Construct the permutation matrix P.

3. Repeat step 1 for $\underline{m} = \underline{u}_i$ with $1 \leq i \leq k$ and obtain the set $\mathcal{P}^{(i)}$. The elements in $\mathcal{P}^{(i)}$ are ordened according permutation matrix P. Take the first element in $\mathcal{P}^{(0)}$ and subtract it from the first element in $\mathcal{P}^{(i)}$ to obtain the i-th row $\underline{e}_i$ of the encipher matrix.

4. Construct the encipher matrix $E = (\underline{e}_1^T, \underline{e}_2^T, \ldots, \underline{e}_k^T)^T$, calculate the decipher matrix D, the matrix E^{-1} and construct the syndrome-error table.

<u>Costs</u>

$k\theta(N \log N)$ enciphermTents on average,

$kO(n \log N)$ operations for ordening,

where

$N = n-i+1$ for a non-cyclic code.

<u>Remark</u>. The assumption that messages of the kind $\underline{m}=\underline{0}$ and $\underline{m}=\underline{u}_i$ are not allowed to obtain an increased security [6, p. 40] is merely outward

seeming. In the above attack it is not necessary to use the messages $\underline{m}=\underline{0}$ and $\underline{m}=\underline{u}_i$. One can take an arbitrary message $\underline{m}$. The unit vector $\underline{u}_i$ can be constructed by choosing a message $\underline{m}_i$ such that $\underline{m}_i = \underline{m} + \underline{u}_i$. The additional costs involved are k times a k-dimensional vector addition over $\mathbb{F}_q$ which can be neglected.

5. Attack by Struik

Hin's attack is based on the imposed structure on the error patterns. Struik's [8] generalized version of Hin's attack does not assume any structure about the error patterns and is also applicable to the generalized Rao-Nam scheme as outlined in section 2.

An error pattern $\underline{z}$ is randomly selected from the set $\mathcal{Z} = \{\underline{z}^{(j)}\}_{j=1}^{N}$ which contains N different error patterns. After encipherment the error pattern is permuted; the set $\mathcal{Z}^P$ is defined as $\{\underline{z}^{(j)}P\}_{j=1}^{N}$. From the set $\mathcal{Z}$ we can define a set $\mathcal{Z}_\Delta$ of error pattern differences $\underline{z}^{(i,j)} = \underline{z}^{(i)}-\underline{z}^{(j)}$ and in the same way the set $\mathcal{Z}_\Delta^P = \{\underline{z}^{(i,j)}P\}_{i,j=1}^{N}$ can be obtained. A guessed error pattern is denoted by $\hat{\underline{z}}$ and the difference $\tilde{\underline{z}}^{(i,j)} = \hat{\underline{z}}^{(i)}-\underline{z}^{(j)}$. Because there are N distinct error patterns, there are N distinct permuted error patterns. For an arbitrary message $\underline{m}$ there are N distinct encipherments $\underline{c}^{(j)} = \underline{m}E+\underline{z}^{(j)}P$ possible, which will be denoted by the set $\mathcal{C} = \{\underline{c}^{(j)}\}_{j=1}^{N}$. For the construction of a unit vector $\underline{u}_i$ we choose a message $\underline{m}_i$ such that $\underline{m}_i = \underline{m}+\underline{u}_i$. The set of encipherments of message $\underline{m}_i$ is denoted by $\mathcal{C}_i$.

The attack can now be described as follows. Encipher an arbitrary message $\underline{m}$ until all the N different cryptograms $\underline{c}^{(1)}, \underline{c}^{(2)}, \ldots, \underline{c}^{(N)}$ are obtained. Construct with the N different encipherments a directed labeled graph $\Gamma=(\mathcal{C},\mathcal{Z}_\Delta^P)$. The vertices are $\underline{c}^{(1)}, \underline{c}^{(2)}, \ldots, \underline{c}^{(N)}$ and the edge from vertex $\underline{c}^{(i)}$ to vertex $\underline{c}^{(j)}$ has label $\underline{z}^{(i,j)}P$. The relation follows from $\underline{c}^{(i)}-\underline{c}^{(j)} = (\underline{m}E+\underline{z}^{(i)}P) - (\underline{m}E+\underline{z}^{(j)}P) = \underline{z}^{(i,j)}P$. In the binary case (q=2) it follows that $\underline{z}^{(i,j)}P = \underline{z}^{(j,i)}P$. Subsequently construct the automorphism group Aut(Γ), i.e. all the permutations on $\mathcal{C}$ which leave all the labels $\underline{z}^{(i,j)}P$ invariant.

We choose a message $\underline{m}_i = \underline{m}+\underline{u}_i$. Again we encipher until all the N different cryptograms $\underline{c}_i^{(j)}$ are obtained. Subsequently the graph $\Gamma_i=(\mathcal{C}_i,\mathcal{Z}_\Delta^P)$ is constructed. Select at random one automorphism ϕ from the automorphism group Aut(Γ). The mapping induced by ϕ from Γ_i on Γ gives N cryptograms $\underline{c}_i^{(1)}$, $\underline{c}_i^{(2)}$, .. ,$\underline{c}_i^{(N)}$ synchronized in a certain way with $\underline{c}^{(1)}$, $\underline{c}^{(2)}$, .. ,$\underline{c}^{(N)}$. Calculate $\underline{c}_i^{(1)}-\underline{c}^{(1)} = (\underline{m}_i E+\hat{\underline{z}}_i^{(1)}P) - (\underline{m}E+\underline{z}^{(1)}P) = \underline{e}_i+\tilde{\underline{z}}^{(1,1)}P$. With probability $|\mathrm{Aut}(\Gamma)|^{-1}$ the row $\underline{e}_i$ will be correctly estimated as there exists an automorphism ϕ for which $\tilde{\underline{z}}^{(1,1)}= 0$. The correctness of a row can not be verified independently from the other rows. On average we can expect that the cryptanalyst has to construct $|\mathrm{Aut}(\Gamma)|^k$ encipher matrices $\hat{E}$ before the correct one is obtained, calculate the decipher matrix $\hat{D}$, the matrix $\hat{E}^{-1}$ and construct the syndrome-error table via $\underline{c}^{(j)}-\underline{m}E = \underline{z}^{(j)}P$. In this way an (equivalent) decoding algorithm is obtained.

SUMMARY

1. Encipher a message $\underline{m}$ until all the N different cryptograms $\underline{c}^{(1)}$, $\underline{c}^{(2)}$, ... , $\underline{c}^{(N)}$ are obtained.

2. Construct the directed complete graph $\Gamma=(\mathcal{V},\mathcal{Z}_\Delta^P)$ and the automorphism group Aut(Γ).

3. For $1\le i\le k$, choose a message $\underline{m}_i$ such that $\underline{m}_i = \underline{m}+\underline{u}_i$. Repeat step 1 for $\underline{m} = \underline{m}_i$ and construct $\Gamma_i=(\mathcal{V}_i,\mathcal{Z}_\Delta^P)$.

4. For $1\le i\le k$ select at random a automorphism ϕ from the automorphism group Aut(Γ). Map Γ_i on Γ according ϕ and calculate $\hat{\underline{e}}_i = \underline{c}_i-\underline{c}$.

5. Construct the encipher matrix $\hat{E} = (\hat{\underline{e}}_1^T, \hat{\underline{e}}_2^T, \ldots, \hat{\underline{e}}_k^T)^T$, calculate the decipher matrix $\hat{D}$, the matrix $\hat{E}^{-1}$ construct the syndrome-error table and verify the solution. If the solution is not correct, repeat the steps 4 and 5.

Costs

Preliminary work

$k\Theta(N \operatorname{Log} N)$ encipherments on average,

$O((k+|\mathrm{Aut}(\Gamma)|)\ n\ \lceil\log q\rceil)$ bits of memory,

$O(nN\lceil\log q\rceil)$ bits of temporal memory space,

$O(knN^2 \log N)$ operations.

Calculation of encipher matrix

$O(kn\ \lceil\operatorname{Log} q\rceil)$ bits,

$O(kn|\mathrm{Aut}(\Gamma)|^k)$ operations.

Validation costs are neglected.

From the costs it appears that the number of possible automorphisms $|\mathrm{Aut}(\Gamma)|$ is a measure of the computational strength of the (generalized) Rao-Nam scheme. In the worst case situation where $|\mathrm{Aut}(\Gamma)| = N$, the costs of this attack is the same as those of the attack described by Rao and Nam. There exist

$$q^{k(n-k+1)}$$

combination of error patterns each chosen from a distinct coset such that the upper bound

$$|\mathrm{Aut}(\Gamma)| = q^{n-k}$$

is reached. From this result it follows that the probability of selecting the right combination randomly that leads to the maximum number of automorphisms is approximately q^{-kN}. For this reason the function that determines this set of error patterns must be highly structured and is also part of the secret key.

In the Rao-Nam scheme using ATE's the number of automorphisms ϕ from the group $\mathrm{Aut}(\Gamma)$ if $N>2$ is given by:

$|\mathrm{Aut}(\Gamma)| = 2$, iff the ATE's are cyclic and have Hamming weight $\frac{n}{2}$,
$|\mathrm{Aut}(\Gamma)| = 1$, else.

If $|\mathrm{Aut}(\Gamma)| = 1$ then the automorphism is: $\phi(\underline{z}) = \underline{z}$ and,
if $|\mathrm{Aut}(\Gamma)| = 2$ then we have also $\phi(\underline{z}) = \underline{z} + \underline{1}$.

We can conclude that the Rao-Nam scheme is insecure against a chosen-plaintext attack in many cases. Only if the value $|\mathrm{Aut}(\Gamma)|$ is large this attack wil not work. In the next section an attack is given which is efficient when the number of automorphisms $|\mathrm{Aut}(\Gamma)|$ is large.

<u>Remark</u>. In theory this attack can be applied to the McEliece scheme too. In this case the order of the automorphism group is one and contains the identity automorphism only. However the N possible error patterns is astronomically large. Therefore this attack fails due to

the large amount of preliminary work involved.

6. EXTENDED ATTACK BY STRUIK

In the following attack an estimated generator matrix is used to perform a ciphertext-only attack. The attack can be divided into two parts. The first part is based on the chosen-plaintext attack to obtain an estimated generator matrix $\hat{E}$. The second part is a ciphertext-only attack from which the unkown message $\hat{\underline{m}}$ is guessed. With this attack we do not obtain the correct generator matrix, however we do obtain the unknown message.

FIRST PART - Chosen-Plaintext Attack

The first part of the attack is the same as described in section 5. We stop after the first estimate of the generator matrix E. The guessed generator matrix is denoted by $\hat{E}$, where $\hat{\underline{e}}_i = \underline{e}_i + \tilde{\underline{z}}^{(i,i)}P$. Since each automorphism is a translation, the rows must be equal to $\hat{\underline{e}}_i = \underline{e}_i + \underline{v}_i$ for a certain unkown $\underline{v}_i \in V$, which is a sub-space of $(\mathbb{F}_q)^n$.

SECOND PART - Ciphertext-Only Attack

Let $\hat{\underline{c}}$ denote the encipherment $\hat{\underline{m}}E + \hat{\underline{z}}P$ of the unknown message $\hat{\underline{m}}$.

$$\tilde{\underline{c}} = \hat{\underline{c}} - \underline{c} = (\hat{\underline{m}} - \underline{m})E + (\hat{\underline{z}} - \underline{z})P = \tilde{\underline{m}}E + \tilde{\underline{z}}P ,$$

$$\text{where } \tilde{\underline{m}} = \hat{\underline{m}} - \underline{m} \text{ and } \tilde{\underline{z}} = \hat{\underline{z}} - \underline{z} .$$

Since $\hat{\underline{e}}_i = \underline{e}_i + \underline{v}_i$ it follows that

$$\tilde{\underline{c}} = \tilde{\underline{m}}\,\hat{E} + \tilde{\underline{z}}P - \sum_{i=1}^{k} \tilde{m}_i \underline{v}_i$$

If the cryptanalyst knows $\tilde{\underline{z}}P - \sum \tilde{m}_i \underline{v}_i$ then he can solve $\tilde{\underline{m}}$ from the above equation. The cryptanalyst picks at random a vector $\underline{w} \in V$ and calculates:

$$\underline{a} = \tilde{\underline{c}} - \underline{w} = \tilde{\underline{m}}\hat{E} + (\tilde{z}P - \sum_{i=1}^{k} \tilde{m}_i \underline{v}_i - \underline{w})$$

Suppose $\tilde{\underline{z}}P - \sum \tilde{m}_i \underline{v}_i - \underline{w} = \underline{0}$, then $\tilde{\underline{m}}$ can be solved from $\tilde{\underline{m}}\hat{E} = \underline{a}$. Repeat until a sensible plaintext $\underline{m} + \tilde{\underline{m}}$ is obtained. Note that $\forall\ \underline{z}\ \exists\ \underline{w}\ [\ \underline{z}\tilde{P} - \sum m_i \tilde{\underline{v}}_i - \underline{w} = \underline{0}\ ]$. The expected number of attempts is at most N.

Costs

$O(k^2nN)$ operations,

$O(kn\lceil\log q\rceil)$ bits memory space.

A refinement of this attack will be given in a paper to apear.

7. Modified Scheme

Almost all the proposed attacks on the Rao-Nam scheme are based on estimating the rows of the encipher matrix E=SGP by constructing unit vectors or by solving a system of linear equations. Therefore to avoid such attacks the S-matrix should be replaced by a non-linear function. In general S can be replaced by a secret invertible function f which transforms a message $\underline{m} \in (\mathbb{F}_q)^k$ into a word $\underline{m}' \in (\mathbb{F}_q)^k$. As special case this function may depend on the error vector $\underline{z}$ used too, as we can determine $\underline{z}$ from the cryptogram in a unique way. In this case the following enciphering scheme is obtained:

$$\underline{c} = f(\underline{m},\underline{z}).E + \underline{z},$$

where E = GP and f chosen such that

$$\forall \underline{z}\ \forall \underline{m} \quad f^{-1}(f(\underline{m},\underline{z}),\underline{z}) = \underline{m}.$$

The decoding algorithm is almost the same as described before if f^{-1} is used instead of S^{-1} and is as follows.

1) Determine $\underline{c}^* = \underline{c}D^T$ and obtain the error pattern $\underline{z} \in (\mathbb{F}_q)^n$.
   As result $\underline{c}' = \underline{c} - \underline{z} = f(\underline{m},\underline{z})E$ is obtained.
3) Calculate $\underline{m}' = \underline{c}'E^{-R} = f(\underline{m},\underline{z})$, in which E^{-R} is a right inverse of the matrix E.
4) The final result is the plaintext $\underline{m} = f^{-1}(\underline{m}',\underline{z})$.

This scheme is not vulnerable to chosen-plaintext attacks as described. This can be seen easily from the fact that the secret function f can be chosen such that it does not allow construction of unit vectors to estimate a row in the E=GP matrix. Hence this scheme provides a higher security level.

8. Conclusion

The Rao–Nam scheme is generalized to $\mathbb{F}_q$. For this general scheme 3 chosen-plaintext attacks are discussed.

It is shown that the Rao–Nam scheme using ATE's is completely insecure against a chosen-plaintext attack. If a pre-defined set of error vectors is used then it appears that the number of possible automorphisms $|\mathrm{Aut}(\Gamma)|$ is a measure for the computational strength of the (generalized) Rao–Nam scheme. If the value $|\mathrm{Aut}(\Gamma)|$ is small then the attack described in section 5 is efficient, otherwise the attack described in section 6 will break the scheme.

We have given an improved scheme in which the linear transformation S in the encipher matrix E=SGP is replaced by a non-linear funtion. This improved scheme is not vulnerable to the chosen-plaintext attacks as described.

ACKNOWLEDGEMENT
The authors thank J.P. Boly, P.J.M. Hin, H. Meijer and H.C.A. van Tilborg for helpful conversations.

REFERENCES

[1] **McEliece, R.J.**, "A Public-Key Cryptosystem Based On Algebraic Coding Theory", DSN Progress Report 42-44, Pasadena, JPL, pp.114-116, 1978.

[2] **Berlekamp, E.R., McEliece, R.J, and van Tilborg, H.C.A,** "On the Inherent Intractability of Certain Coding Problems", IEEE Trans. Inform. Theory. IT-24, pp.384-386, 1978.

[3] **Adams, C, Meijer, H,** "Security relating comments regarding the McEliece Public-Key Cryptosystem, presented at crypto'87.

[4] **Jorrissen, F,** "A Security Evaluation of the Public-Key Cipher System Proposed by R.J. McEliece, used as Combined Scheme", Katholieke Universiteit Leuven, Lab. ESAT, 1986.

[5] **Jordan, J.P.,** "A Variant of a Public Key Cryptosystem based on Goppa Codes", Sigact news, vol 15, no: 1, pp. 61-66, 1983.

[6] **Rao, T.R.N., Nam, K.H.,** "Private-Key Algebraic-Coded Cryptosystem", in: Advances in Cryptology - CRYPTO'86, A.M. Odlyzko (Ed.), Lecture Notes in Computer Science #263, Springer, pp 35-48, 1987.

[7] **Hin, P.J.M.,**"Channel-Error-Correcting Privacy Cryptosystems", Thesis, Delft Univ. of Techn., 1986 (in Dutch).

[8] **Struik, R.,** "Algebraic Coded Cryptosystems", Private Communication, July 1987.

[9] **van Tilburg, J.,** "Private-Key Cryptosystems based on Algebraic Coding Theory", Pub 87 DNL/53, PTT/DNL, the Netherlands, 1987.

ON STRUIK-TILBURG CRYPTANALYSIS OF RAO-NAM SCHEME

T.R.N Rao
The Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, Louisiana 70504

Introduction

A private-key cryptosystem using algebraic codes was presented in CRYPTO 86 [1] and it is referred later [2] and also here as Rao-Nam Scheme. Subsequently, a chosen-plaintext attack was presented by Struik and Tilburg in a rump session of CRYPTO 87 and appears in this issue [2]. This note addresses a major difference between the types of chosen-plaintext attacks on private-key algebraic code cryptosystems vs. the other more conventional private key schemes, and presents a rebuttal to the conclusions given in [2].

Chosen-Plaintext Attacks

In a conventional private-key cryptosystem, the ciphertext for a given message, M with a key K is given by

$$C = E_k(M) .$$

The ciphertext C is a constant for any given M and K. However, in algebraic-code cryptosystem as given in [1]

$$C = MG'' + Z',$$

where G'' is a matrix combinationally equivalent to generator matrix G of a Linear Code and Z' is a selected error vector Z appropriately permuted by a secret function P. See details [1].

Since Z is a randomly selected error vector, it is conceivable that for a given M and encryption algorithm, many ciphertexts say $C_1, C_2, \cdots$ can be obtained. This is a major difference and it can be exploited in a chosen-plaintext attack as follows. The cryptanalyst would obtain all possible ciphertexts $C_1, C_2, \cdots$ for a given M (under the same encryption key) in an exhaustive manner. This will enable him to draw information on the error vectors $Z_1, Z_2, \cdots$ which then are used to break the code. This line of attack was actually suggested by Rao-Nam [1] and was indeed pursued rather vigorously by Struik-Tilburg [2]. However, the basic contention would be on "how many different ciphertexts are indeed required for one particular M in order to break code ?" In other words, "what are these numbers (in any specified environment of network of users) to designate a scheme to be secure or unsecure ?" Without some consideration for these numbers it would be meaningless to designate a scheme as "weak" or as "insecure". We take Struik-Tilburg analysis to determine these numbers as follows.

Struik-Tilburg Analysis [2] and Discussion

In order to obtain information on Z's sufficient to break the code we obtain all different ciphertexts $C_1, C_2, \cdots, C_N$ exhaustively for a given plaintext M, where N is the number of all distinct error vectors. The expected number of attempts to obtain all distinct C_i are shown to be $N\ln N$. (Struik-Tilburg give $N\log N$ in their rump session paper.) Since this procedure is to be repeated for the k unit vectors $(u_i\ , u_i=1,2,\cdots,k)$ the total number of ciphertexts required for chosen plaintexts is $O(kN\log N)$. Struik-Tilburg analyze as follows. For the ATE method of Rao-Nam Scheme, since $N \leq n$, This number is rather small and hence the scheme is easily broken. Also for syndrome-error table method (of Rao-Nam Scheme) using codes with high information rates, $N=2^{n-k}$ is also small enough that the scheme is not very secure according to [2]. We show below by a consideration of two examples that this analysis is flawed and that the conclusions of [2] are wrong.

The argument we advance here in this paper is as follows. In any practical network operations, for any chosen-plaintext M under a specific key it would be impossible to obtain thousands of different ciphertexts. For example, if the network administrator is distributing a specific bulletin or message to all users then a common key may be used but then the same ciphertext will be distributed to all users. Alternately, secret keys may be used individually for each user. In either of these scenarios, there is no way a large number (thousands) ciphertexts for any one particular bulletin will be made available. For more frequently used blocks of message items it is conceivable for the cryptanalyst to gather some number of ciphertexts but it would be near impossible to obtain tens of thousands of ciphertexts as required by analysis of [2]. The code examples below exlain these numbers required for analysis.

Examples

Consider (72, 64, 4) Hamming Code with Z-errors selected randomly from the standard-array table as suggested in [1]. For this case the $N=2^{n-k}=2^8$ and $N\log N$ is 2024, and $kN\log N=129{,}000$. Consider as another example, a (32, 28, 5) Reed-Solomon Code over GF(2^8) with information ratio IR=28/32=.875. The corresponding parameters in GF(2) are $n=32\times8=256$, $k=28\times8=224$ and $n-k=32$. For this code the number of different Z-vectors are $N=2^{32}$ and $N\log N=2^{37}$, and $kN\log N\approx2^{46}$. These figures are so large that any chosen-plaintext attack along the lines of [2] is practically impossible.

Conclusion

This discussion and the examples show clearly that Rao-Nam scheme appears very secure under chosen-plaintext attacks of this type. There is more to be encouraged about the security of the scheme after this cryptoanalysis.

References

[1] T.R.N. Rao and K.H. Nam, "Private-Key Algebraic-Code Cryptosystems", Lecture Notes in Computer Science, Advances in Cryptology-CRYPTO '86, pp. 35-48, (Editor A.M. Odlyzko), Springer-Verlag 1987.

[2] R. Struik and J. Van Tilburg, "The Rao-Nam Scheme is insecure against a chosen-plaintext attack", A Rump Session paper, CRYPTO '87. A revised version appears in this issue.

A Generalization of Hellman's Extension of Shannon's Approach to Cryptography

(Abstract)

Pierre Beauchemin
Gilles Brassard

Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, Succursale "A"
Montréal, Québec
Canada H3C 3J7

In his landmark 1977 paper [Hell77], Hellman extends the Shannon theory approach to cryptography [Shan49]. In particular, he shows that the expected number of spurious key decipherements on length n messages is at least $2^{H(K)-nD}-1$ for *any* uniquely encipherable, uniquely decipherable cipher, as long as each key is equally likely and the set of meaningful cleartext messages follows a uniform distribution (where $H(K)$ is the key entropy and D is the redundancy of the source language). In this paper, we show that Hellman's result holds with no restrictions on the distribution of keys and messages. We also bound from above and below the key equivocation upon seeing the ciphertext. Sufficient conditions for these bounds to be tight are given. The results are obtained through very simple purely information theoretic arguments, with no needs for (explicit) counting arguments.

The formal statements and proofs will be provided in the final paper, to appear in the *Journal of Cryptology* [BB88].

Bibliography

[BB88] Beauchemin, P. and G. Brassard, "A generalization of Hellman's extension of Shannon's approach to cryptography", to appear in *Journal of Cryptology*, 1988.

[Hell77] Hellman, M. E., "An extension of the Shannon theory approach to cryptography", *IEEE Transactions on Information Theory*, vol. IT-23, 1977, pp. 289-294.

[Shan49] Shannon, C. E., "Communication theory of secrecy systems", *Bell System Technical Journal*, vol. 28, 1949, pp. 656-715.

MULTIPARTY UNCONDITIONALLY SECURE PROTOCOLS

(Abstract)

David Chaum[†]
Claude Crépeau[‡]
Ivan Damgård[*]

[†]*Centre for Mathematics and Computer Science*
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

[‡]*Laboratory for Computer Science, M.I.T.*
545 Technology Square, Cambridge, MA 02139, U.S.A.

[*]*Matematisk Institut, Aarhus Universitet,*
Ny Munkegade, DK 8000 Aarhus C, Danmark

Abstract

It has been shown previously how almost any multiparty protocol problem can be solved. All the constructions suggested so far rely on trapdoor one-way functions, and therefore must assume essentially that public key cryptography is possible. It has also been shown that unconditional protection of a single designated participant is all that can be achieved under that model. Assuming only authenticated secrecy channels between pairs of participants, we show that essentially any multiparty protocol problem can be solved. Such a model actually implies the further requirement that less than one third of the participants deviate from the protocol. The techniques presented do not, however, rely on any cryptographic assumptions; they achieve the optimal result and provide security as good as the secrecy and authentication of the channels used. Moreover, the constructions have a built-in fault tolerance: once the participants have sent messages committing themselves to the secrets they will use in the protocol, there is no way less than a third of them can stop the protocol from completing correctly. Our technique relies on the so called key-safeguarding or secret-sharing schemes proposed by Blakley and Shamir as basic building blocks. The usefulness of their homomorphic structure was observed by Benaloh, who proposed techniques very similar to ours.

AUTHOR INDEX